

**RANCANG BANGUN GENERATOR SET UJI UNTUK DIAGNOSIS
KESALAHAN RANGKAIAN *MULTIPLEXER QUAD 2 LINE TO 1 LINE* IC
74157 MENGGUNAKAN METODE TABEL KESALAHAN BERBASIS
ARDUINO NANO ATMEGA328**

(Tesis)



Oleh

RIZKIMA AKBAR SETIAWAN

**PROGRAM PASCASARJANA MAGISTER TEKNIK ELEKTRO
FAKULTAS TEKNIK
UNIVERSITAS LAMPUNG
BANDAR LAMPUNG
2023**

**RANCANG BANGUN GENERATOR SET UJI UNTUK DIAGNOSIS
KESALAHAN RANGKAIAN *MULTIPLEXER QUAD 2 LINE TO 1 LINE* IC
74157 MENGGUNAKAN METODE TABEL KESALAHAN BERBASIS
ARDUINO NANO ATMEGA328**

**Oleh
RIZKIMA AKBAR SETIAWAN**

Tesis

**Sebagai Salah Satu Syarat untuk Mencapai Gelar
MAGISTER TEKNIK ELEKTRO**

Pada

**Program Pascasarjana Magister Teknik Elektro
Fakultas Teknik Universitas Lampung**



**PROGRAM PASCASARJANA MAGISTER TEKNIK ELEKTRO
FAKULTAS TEKNIK
UNIVERSITAS LAMPUNG
BANDAR LAMPUNG
2023**

ABSTRACT

DESIGN SET-TEST GENERATOR FOR FAULT DIAGNOSIS OF MULTIPLEXER QUAD 2 LINE TO 1 LINE IC 74157 USING FAULT TABLE METHOD ON ARDUINO NANO ATMEGA238-BASED

By

RIZKIMA AKBAR SETIAWAN

A combinational digital circuit is composed of several logic gates whose output conditions are caused by the input conditions of the circuit. Minor faults in combinational digital circuits become a fundamental problem. These minor faults will cause temporary or permanent disturbances to the main output if they are not detected. This study aims to build a test generator set that successfully detects faults and locations in the quad 2-line to 1-line IC 74157 multiplexer circuit. The fault table method is used to compile a fault diagnosis tree as a test set embedded in a test generator set. The complete generator set test consists of 4 generator set test pieces of the multiplexer quad 2-lines to 1-line IC 74157. Each part of the circuit consists of 4 inputs with the Boolean equation $Y = \overline{A}C\overline{D} + \overline{A}BD$ which consists of 11 lines. The test results on the IC 74157 quad 2-line to 1-line multiplexer circuit section provide an efficiency of 68.75% - 81.25%, with the most extended test being 5 levels. Complete testing of the entire quad 2-line to 1-line multiplexer IC 74157 gives an efficiency of 98.05% - 98.83%, with the most extended test being 20 levels.

Keyword: Fault, Multiplexer quad 2-line to 1-line IC 74157, Faults Table Method, Set-test Generator.

ABSTRAK

RANCANG BANGUN GENERATOR SET UJI UNTUK DIAGNOSIS KESALAHAN RANGKAIAN MULTIPLEXER *QUAD 2 LINE TO 1 LINE* IC 74157 MENGGUNAKAN METODE TABEL KESALAHAN BERBASIS ARDUINO NANO ATMEGA238

Oleh

RIZKIMA AKBAR SETIAWAN

Rangkaian digital kombinasional merupakan rangkaian yang tersusun dari beberapa gerbang logika dengan kondisi keluaran disebabkan oleh kondisi masukan rangkaian tersebut. Kesalahan kecil pada rangkaian digital kombinasional menjadi masalah penting. Jika kesalahan kecil tersebut tidak terdeteksi maka akan menyebabkan gangguan sementara bahkan permanen pada keluaran utamanya. Penelitian ini bertujuan untuk membangun generator set uji yang berhasil mendeteksi kesalahan dan lokasi kesalahan pada rangkaian *multiplexer quad 2 line to 1 line* IC 74157. Metode tabel kesalahan digunakan untuk menyusun pohon diagnosa kesalahan sebagai set pengujian yang ditanamkan pada generator set uji. Generator set uji lengkap terdiri dari 4 buah generator set uji potongan rangkaian *multiplexer quad 2 line to 1 line* IC 74157. Setiap potongan rangkaian terdiri dari 4 *input* dengan persamaan Boolean $Y = \overline{A}C\overline{D} + \overline{A}BD$ yang terdiri dari 11 jalur. Hasil pengujian pada potongan rangkaian *multiplexer quad 2 line to 1 line* IC 74157 memberikan efisiensi sebesar 68,75% - 81,25% dengan pengujian terpanjang adalah 5 level. Pengujian lengkap keseluruhan rangkaian *multiplexer quad 2 line to 1 line* IC 74157 memberikan efisiensi sebesar 98,05% - 98,83% dengan pengujian terpanjang adalah 20 level.

Kata Kunci: Kesalahan, *Multiplexer quad 2 line to 1 line* IC 74157, Metode Tabel Kesalahan, Generator Set Uji.

Judul Tesis : **RANCANG BANGUN GENERATOR SET UJI
UNTUK DIAGNOSIS KESALAHAN RANGKAIAN
MULTIPLEXER QUAD 2 LINE TO 1 LINE IC
74157 MENGGUNAKAN METODE TABEL
KESALAHAN BERBASIS ARDUINO NANO
ATMEGA238**

Nama Mahasiswa : **Rizkima Akbar Setiawan**

Nomor Pokok Mahasiswa : 2025031010

Program Studi : Magister Teknik Elektro

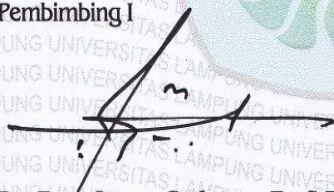

Fakultas : Teknik




1. Komisi Pembimbing

Pembimbing I

Pembimbing II

 **Dr. Eng. Ageng Sadnowo R., S.T., M.T.**  **Dr. Eng. F.K. Arinto S., S.T., M.T.**
NIP 19690228 199803 1 001 NIP 19691219 199903 1 002

2. Ketua Program Studi Magister Teknik Elektro


Misfa Susanto, S.T., M.T., Ph.D.
NIP 19710525 199903 1 001

MENGESAHKAN

1. Komisi Penguji

**Ketua Komisi Penguji
(Pembimbing I)**

: Dr. Eng. Ageng S. R., S.T., M.T.

**Sekretaris Komisi Penguji
(Pembimbing II)**

: Dr. Eng. F.X. Arinto S., S.T., M.T.

**Anggota Komisi Penguji
(Pengujii I)**

: Dr. Ir. Sri Ratna Sulistiyanti, M.T.

**Anggota Komisi Penguji
(Pengujii II)**

: Dr. Sri Purwiyanti, S.T., M.T.

2. Dekan Fakultas Teknik

Dr. Eng. Ir. Helmy Fitriawan, S.T., M.Sc.
NIP 19750928 200112 1 002

3. Direktur Program Pascasarjana

Prof. Dr. Ir. Murhadi, M.Si.
NIP 19640326 198902 1 001

Tanggal Lulus Ujian Tesis : 22 Juni 2023

SURAT PERNYATAAN

Dengan ini Saya menyatakan bahwa sesungguhnya tesis yang saya susun sebagai syarat untuk mendapatkan gelar Magister Teknik pada Program Pascasarjana Magister Teknik Elektro seluruhnya adalah benar merupakan hasil karya sendiri.

Adapun bagian-bagian tertentu dalam penulisan tesis ini, saya kutip dari hasil penulisan orang lain yang sumbernya dituliskan dengan jelas sesuai dengan norma, kaidah dan etika penulisan karya ilmiah.

Tesis dengan judul "Rancang Bangun Generator Set Uji untuk Diagnosis Kesalahan Rangkaian *Multiplexer Quad 2 Line to 1 Line* IC 74157 Menggunakan Metode Tabel Kesalahan Berbasis Arduino Nano Atmega328" dapat diselesaikan berkat bimbingan dan motivasi dari pembimbing-pembimbing saya, yaitu:

1. Dr. Eng. Ageng Sadnowo Repelianto, S.T., M.T.
2. Dr. Eng. F.X. Arinto Setyawan, S.T., M.T.

Saya ucapkan terima kasih yang sebesar-besarnya kepada semua pihak, khususnya kedua dosen pembimbing dan Bapak/ Ibu Dosen Program Studi Magister Teknik Elektro Universitas Lampung yang telah banyak memberikan ilmu pengetahuan, bimbingan dan motivasi.

Apabila dikemudian hari ditemukan seluruh atau sebagian tesis yang saya buat ini bukan hasil karya saya sendiri atau adanya plagiat dalam bagian-bagian tertentu, saya bersedia menerima sanksi akademik sesuai dengan peraturan perundangan yang berlaku.

Bandar Lampung, 26 Juni 2023



Rizkima Akbar Setiawan
NPM: 2025031010

RIWAYAT HIDUP



Penulis dilahirkan di Pringsewu, Lampung, pada tanggal 15 Maret 1997. Penulis merupakan anak tunggal, dari pasangan Bapak Bambang Hariyanto, S.H dan Ibu Fatimah S.E.

Penulis pertama kali mengenyam pendidikan di TK Aisyah Ambarawa. Penulis melanjutkan tingkat sekolah dasar di SD Negeri 1 Margodadi Kec. Ambarawa lulus tahun 2009. Sekolah Menengah Pertama diselesaikan di SMP Negeri 1 Pringsewu Kec.

Pringsewu, lulus tahun 2012. Sekolah Menengah Atas diselesaikan di SMA Negeri 1 Gadingrejo Kec. Gadingrejo, lulus tahun 2015. Penulis melanjutkan pendidikan ke jenjang perguruan tinggi di Universitas Lampung pada Jurusan Teknik Elektro tahun 2015 melalui jalur Mandiri (UM).

Selama menimba ilmu di Universitas Lampung penulis aktif dalam organisasi Badan Mahasiswa Pringsewu Seluruh Indonesia (BMP-SI) menyandang jabatan sebagai Menteri Pemberdayaan Lingkungan pada tahun 2017.

Pada Tahun 2018, Penulis melaksanakan Kerja Praktik (KP) di PTPN VII Unit Usaha Way Berulu Pesawaran selama satu bulan. Penulis menyelesaikan Kerja Praktik dengan menulis sebuah laporan yang berjudul: “Sistem *Control* Suhu Ruang Menggunakan *Resistance Temperature Detector PT100 3Wire* pada *Oven* Pengasapan Karet RSS (*Ribbed Smoked Sheets*) di PT. Perkebunan Nusantara VII Unit Usaha Way Berulu”.

Pada Tahun 2019, Penulis menyelesaikan program studi S1 Teknik Elektro dengan menulis sebuah skripsi yang berjudul: “Perancangan Setrika Listrik Tanpa Kabel dengan Pengaturan Suhu Otomatis Berbasis Arduino Uno”.

Pada Tahun 2020 penulis terdaftar sebagai Mahasiswa Program Pascasarjana Teknik Elektro Universitas Lampung melalui jalur Beasiswa bebas SPP. Dan pada tahun 2022 penulis melakukan penelitian pada bidang mikrokontroler dengan judul tesis “Rancang Bangun Generator Set Uji untuk Diagnosis Kesalahan Rangkaian *Multiplexer Quad 2 Line to 1 Line* IC 74157 Menggunakan Metode Tabel Kesalahan Berbasis Arduino Nano Atmega328” dibawah bimbingan Bapak Dr. Eng. Ageng Sadnowo Repelianto, S.T., M.T. dan Bapak Dr. Eng. F.X. Arinto Setyawan, S.T., M.T

Bandar Lampung, 26 Juni 2023
Penulis

RIZKIMA AKBAR SETIAWAN



PERSEMBAHAN

Dengan Ridho Allah SWT, teriring shalawat kepada Nabi Muhammad SAW

Karya Tulis ini kupersembahkan untuk:

Ayah dan Ibuku Tercinta

Bambang Hariyanto, S.H. dan Fatimah, S.E.

Keluarga Tersayang

*Keluarga Besar H. S. Markam dan Keluarga Besar Pangeran Usman
serta Keluarga besarku di Masa Depan*

Dosen Teknik Elektro

*Yang selalu membimbing, mengajarkan, memberikan saran, baik secara akademis
maupun non akademis*

Teman- teman kebanggaanku

Rekan – rekan Jurusan Teknik Elektro

Sahabat-sahabatku

*Yang selalu membantu, memberikan semangat, mendukung menuju keberhasilan,
serta berbagi cerita suka duka dalam berkeluh kesah*

Keluarga Besar Magister Teknik Elektro 2020

*Yang selalu memberi semangat, dukungan dalam proses yang sangat panjang,
dan selalu berdiri bersama dalam perjuangan menuju kesuksesan*

Almamaterku

Universitas Lampung

Bangsa dan Negaraku

Republik Indonesia

Terima kasih untuk semua yang telah diberikan kepadaku. *Jazzakallah Khairan.*



MOTTO

*“Jangan Pernah Berhenti Berbuat Baik untuk Orang Lain Walaupun
Orang Itu akan Menyakitimu, Percayalah Hal Baik akan Selalu
Datang Kepadamu”*

*”Pantang Menyerah Sebelum Dicoba, Pantang Pulang Sebelum
Sukses dan Do’a Ibu adalah Segalanya”*

— Rizkima Akbar Setiawan alias Akbar Tunggal

SAN WACANA

Assalamu'alaikum Wr. Wb.

Syukur Alhamdulillahirobbilalamin, Penulis haturkan puji syukur atas kehadiran Allah SWT yang senantiasa melimpahkan rahmat dan hidayah, serta inayah-Nya kepada penulis sehingga penulis dapat menyelesaikan laporan Tesis dengan mempersembahkan judul tesis **“Rancang Bangun Generator Set Uji untuk Diagnosis Kesalahan Rangkaian Multiplexer Quad 2 Line to 1 Line IC 74157 Menggunakan Metode Tabel Kesalahan Berbasis Arduino Nano Atmega328”** dengan sebaik-baiknya.

Shalawat beriring salam selalu tercurah kepada junjungan seluruh alam Nabi Muhammad SAW, sahabatnya, serta para pengikutnya yang selalu istiqomah diatas jalan agama islam hingga hari ajal menjemput.

Dalam penyusunan tugas akhir ini penulis banyak mendapat bimbingan, motivasi dan bantuan baik moral maupun materi oleh banyak pihak. Untuk itu dengan sepuh ketulusan hati penulis mengucapkan terima kasih kepada:

1. Ibu Prof. Dr. Ir. Lusmeilia Afriani, D.E.A., I.P.M., selaku Rektor Universitas Lampung.
2. Bapak Prof. Dr. Ir. Murhadi, M.Si., selaku Direktur Program Pascasarjana Fakultas Teknik Universitas Lampung.
3. Bapak Dr. Eng. Helmy Fitriawan, S.T., M.Sc., selaku Dekan Fakultas Teknik, Universitas Lampung.
4. Ibu Herlinawati, S.T., M.T., selaku Ketua Jurusan Teknik Elektro Universitas Lampung.

5. Bapak Misfa Susanto, S.T., M.T., Ph.D. selaku Ketua Program Studi Magister Teknik Elektro Universitas Lampung.
6. Bapak Dr. Eng. Ageng Sadnowo Repelianto, S.T., M.T., selaku dosen pembimbing utama tesis sekaligus dosen pembimbing akademik yang banyak memberikan waktu, ide pemikiran dan semangat serta motivasi bagi penulis.
7. Bapak Dr. Eng. F.X. Arinto Setyawan, S.T., M.T., selaku pembimbing kedua tesis, yang telah banyak memberikan waktu, pengalaman, motivasi dan pemikiran bagi penulis.
8. Ibu Dr. Ir. Sri Ratna Sulistiyanti, S.T., M.T., selaku dosen penguji utama sekaligus yang telah banyak memberikan kritik, saran dan motivasi yang bermanfaat bagi penulis.
9. Ibu Dr. Sri Purwiyanti, S.T., M.T., Ph.D., selaku dosen penguji kedua yang telah banyak memberikan kritik dan saran yang bermanfaat bagi penulis.
10. Seluruh Dosen Program Studi Magister Teknik Elektro Universitas Lampung, berkat ilmu yang telah diajarkan kepada penulis selama penulis menjalani masa studi di perkuliahan.
11. Seluruh Tenaga Pendidik Program Studi Magister Teknik Elektro Fakultas Teknik Universitas Lampung yang telah banyak membantu kepada penulis, sehingga dapat menyelesaikan tugas akhir ini.
12. Seluruh teman-teman Program Studi Magister Teknik Elektro Unila angkatan 2020 untuk kebersamaan yang telah dijalani. Tiada kata yang dapat penulis utarakan untuk mengungkapkan perasaan senang dan bangga menjadi bagian dari angkatan 2020.
13. Serta semua pihak yang telah membantu dalam penyusunan tugas akhir ini yang tidak bisa penulis sebutkan satu-persatu.
14. Teman-teman sehidup semati di Teknik Elektro UNILA Kevin, Misbach, Bayu, Hendry, Fajar, Arsy, Catur, Iqbal, Kak Yudi, Mba Vivi, dan Tiya yang telah memberikan semangat juang, serta suasana yang sangat nyaman dalam menempuh gelar M.T.

Akhir kata, Penulis menyadari bahwa Tesis ini masih jauh dari kesempurnaan, baik dari segi isi maupun cara penyajiannya. Oleh karena itu, Penulis sangat mengharapkan saran serta kritik yang bersifat membangun dari pembaca. Akhir kata sedikit harapan penulis semoga karya sederhana ini dapat berguna dan bermanfaat bagi kita semua. Aamiin Allahumma Aamiin.

Wassalamu'alaikum Wr. Wb.

Bandar Lampung, 26 Juni 2023

Penulis,

Rizkima Akbar Setiawan

DAFTAR ISI

	Halaman
HALAMAN JUDULi
ABSTRACTiii
ABSTRAKiv
LEMBAR PENGESAHANv
SURAT PERNYATAANvii
RIWAYAT HIDUPviii
SAN WACANAxii
DAFTAR ISI.....	..xv
DAFTAR TABELxviii
DAFTAR GAMBAR.....	xix
I. PENDAHULUAN	1
1.1. Latar Belakang	1
1.2. Rumusan Masalah	3
1.3. Tujuan Penelitian.....	3
1.4. Manfaat Penelitian.....	3
1.5. Batasan Masalah.....	4
1.6. Hipotesis.....	4
1.7. Sistematika Penulisan.....	4
II. TINJAUAN PUSTAKA.....	6
2.1. Penelitian Terdahulu	6

2.2. Gangguan Digital	11
2.3. Karakteristik Gangguan	11
2.4. Gerbang Logika.....	12
2.4. Rangkaian Kombinasional	13
2.6. Diagnosa Kesalahan	14
2.7. Model Gangguan <i>Stuck</i> Logika.....	14
2.8. Pohon Diagnosa.....	16
III. METODE PENELITIAN	19
3.1. Waktu dan Tempat penelitian	19
3.2. Alat dan Bahan Penelitian	19
3.3. Prosedur Penelitian.....	20
3.4. Metode yang Diusulkan.....	20
3.5. Susunan Rangkaian Pengujian	23
3.6. Metode Tabel Kesalahan.....	25
3.6.1 Penyusunan Tabel Kesalahan.....	25
3.6.2 Penentuan Set Pengujian Lengkap Minimum	33
3.6.3 Pohon Diagnosa	41
3.7. Generator Set Uji.....	46
3.7.1 Perancangan Pengujian	47
3.7.2 Prosedur Generator Set Uji	49
IV. HASIL DAN PEMBAHASAN.....	64
4.1. Hasil Penelitian.....	64
4.1.1 Hasil Generator Set Uji Pertama	64
4.1.2 Hasil Generator Set Uji Kedua.....	65
4.1.3 Hasil Generator Set Uji Ketiga.....	66

4.1.4 Hasil Generator Set Uji Keempat.....	68
4.1.5 Hasil Pengujian Generator Set Uji	68
4.2. Pembahasan.....	69
V. SIMPULAN DAN SARAN	87
5.1. Simpulan.....	87
5.2. Saran.....	87
DAFTAR PUSTAKA	88

DAFTAR TABEL

Tabel	Halaman
2.1. Keluaran Gerbang <i>AND</i> saat Normal dan Dalam Kondisi <i>Stuck</i>	15
2.2. Lokasi Kesalahan Gerbang <i>AND</i> Saat Normal dan Dalam Kondisi <i>Stuck</i>	15
2.3. Pemilihan Set Pengujian Lengkap.....	16
3.1. Tabel Kebenaran IC 74157	25
3.2. Tabel Kesalahan Diagnosa Kesalahan Tunggal Rangkaian 1	26
3.3. Kelompok Kesalahan Potongan Rangkaian Uji Pertama.....	27
3.4. Tabel Kesalahan Diagnosa Kesalahan Tunggal Rangkaian 2	28
3.5. Kelompok Kesalahan Potongan Rangkaian Uji 2	29
3.6. Tabel Kesalahan Diagnosa Kesalahan Tunggal Rangkaian 3	30
3.7. Kelompok Kesalahan Potongan Rangkaian Uji 3	31
3.8. Tabel Kesalahan Diagnosa Kesalahan Tunggal Rangkaian 4.....	32
3.9. Kelompok Kesalahan Potongan Rangkaian Uji 4	33
3.10. Penentuan Set Pengujian Lengkap Minimum	35
4.1. Hasil Pengujian Rangkaian Generator Set Uji Potongan 1 Rangkaian <i>Multiplexer</i> IC 74157	70
4.2. Hasil Pengujian Rangkaian Generator Set Uji Potongan 2 Rangkaian <i>Multiplexer</i> IC 74157	74
4.3. Hasil Pengujian Rangkaian Generator Set Uji Potongan 3 Rangkaian <i>Multiplexer</i> IC 74157	78
4.4. Hasil Pengujian Rangkaian Generator Set Uji Potongan 4 Rangkaian <i>Multiplexer</i> IC 74157	82

DAFTAR GAMBAR

Gambar	Halaman
2.1. Ikatan antara Gangguan, Kesalahan , dan Kegagalan	1
2.2. Karakteristik Gangguan.....	1
2.3. Rangkaian Digital Kombinasional	14
2.4. Konsep dasar Model Gangguan <i>Stuck</i> Logika	14
2.5. Pohon Diagnosa.....	18
3.1. Diagram Alir Perancangan Generator Set Uji.....	21
3.2. Diagram Tulang Ikan Tahapan Penelitian	22
3.3. Rangkaian <i>Multiplexer Quad 2 Line to 1 Line</i> IC 74157.....	23
3.4. Penyederhanaan Rangkaian untuk Penurunan Pengujian	24
3.5. Menghilangkan Baris	34
3.6. Menghilangkan Baris dan Kolom	34
3.7. Proses Penentuan Pengujian Penting Primer.....	37
3.8. Penentuan Pengujian Penting Sekunder (10)	38
3.9. Penentuan Pengujian Penting Sekunder (12)	39
3.10. Penentuan Pengujian Penting Sekunder (4 dan 5)	39
3.11. Penentuan Pengujian Tidak Penting (0).....	40
3.12. Pemilihan Pengujian F^*	41
3.13. (a) Urutan $F^*_{00}(2,10)$, (b) Urutan $F^*_{01}(2,10)$, (c) Urutan $F^*_{10}(2,0)$, (d) Urutan $F^*_{11}(2,0)$ dengan 2 Level Pengujian.....	42
3.14. Urutan Pohon Diagnosa dengan 3 Level Pengujian.....	43
3.15. Urutan Pohon Diagnosa dengan 4 Level Pengujian.....	43
3.16. Urutan Pohon Diagnosa dengan 5 Level Pengujian.....	44
3.17. Pohon Diagnosa Potongan 1 Rangkaian IC 74157	44
3.18. Pohon Diagnosa Potongan 2 Rangkaian IC 74157	45

3.19. Pohon Diagnosa Potongan 3 Rangkaian IC 74157	45
3.20. Pohon Diagnosa Potongan 4 Rangkaian IC 74157	46
3.21. Pengaturan Eksperimen Rangkaian Set Uji	47
3.22. Komponen Sistem Perangkat Lunak.....	47
3.23. Implementasi Potongan Rangkaian Uji Multiplexer.....	48
3.24. Generator Set Uji.....	49
3.25. Hasil Serial Monitor Arduino IDE Rangkaian 1 IC 74157.....	51
3.26. Diagram Alir <i>Coding</i> Arduino Nano Sebagai Generator Set Uji Potongan 1 Rangkaian IC 74157.....	52
3.27. Hasil Serial Monitor Arduino IDE Rangkaian 2 IC 74157.....	53
3.28. Diagram Alir <i>Coding</i> Arduino Nano Sebagai Generator Set Uji Potongan 2 Rangkaian IC 74157.....	54
3.29. Hasil Serial Monitor Arduino IDE Rangkaian 3 IC 74157.....	55
3.30. Diagram Alir <i>Coding</i> Arduino Nano Sebagai Generator Set Uji Potongan 3 Rangkaian IC 74157.....	56
3.31. Hasil Serial Monitor Arduino IDE Rangkaian 4 IC 74157.....	57
3.32. Diagram Alir <i>Coding</i> Arduino Nano Sebagai Generator Set Uji Potongan 4 Rangkaian IC 74157.....	58
3.33. <i>Serial Communication</i>	60
3.34. Tampilan <i>Grid View</i>	61
3.35. Tampilan <i>Text Box Description</i>	62
4.1. GUI Generator Set Uji Pertama	65
4.2. GUI Generator Set Uji Potongan 2 Rangkaian IC 74157	66
4.3. GUI Generator Set Uji Potongan 3 Rangkaian IC 74157.	67
4.4. GUI Generator Set Uji Potongan 4 Rangkaian IC 74157	68
4.5. Grafik Persentase Efisiensi Generator Set Uji 1.....	71
4.6. Grafik Persentase Efisiensi Generator Set Uji 2.....	75
4.7. Grafik Persentase Efisiensi Generator Set Uji 3.....	79
4.8. Grafik Persentase Efisiensi Generator Set Uji 4.....	83

I. PENDAHULUAN

1.1 Latar Belakang

Rangkaian digital kombinasional merupakan rangkaian dengan keadaan keluaranya disebabkan kondisi masukan rangkaian kombinasional tersebut [1]. Rangkaian digital kombinasional tersusun dari beberapa gerbang logika. Rangkaian kombinasional digital yang telah dirancang semestinya bekerja dengan baik sesuai yang diharapkan tanpa terjadi kendala.

Gangguan kecil telah menjadi masalah penting saat ini. Kesalahan yang sering terjadi dapat menyebabkan rangkaian digital kombinasional beroperasi secara tidak baik [2]. Kesalahan biasanya disebabkan oleh gangguan internal dalam jaringan rangkaian tersebut. Jika kesalahan kecil tersebut tidak terdeteksi maka akan menyebabkan gangguan sementara bahkan permanen pada keluaran utamanya [3].

Saat ini gangguan pada rangkaian digital menjadi perhatian utama dilingkup sistem digital *modern*. Dibutuhkan metode baru untuk mengatasi masalah yang ditimbulkan dari kesalahan tersebut yaitu dengan cara mendeteksi tiap kesalahan pada jalur rangkaian digital kombinasional dengan langkah dan waktu seminimal mungkin [4]. Jika terdeteksi kesalahan pada logika kombinasional akan menghasilkan dampak besar pada sistem operasi rangkaian yang harus diatasi dengan metode perbaikan yang tepat. Jenis gangguan yang sering terjadi adalah penyamaran logis yang membatasi propagasi dari letak kesalahan ke keluaran utama. Gangguan tersebut karena *input* gerbang lainnya di sepanjang jalur propagasi menghentikan transisi arus menuju gerbang *output* [3].

Diagnosis merupakan pendeteksian untuk mencari kesalahan dan letak pasti dari kesalahan yang terjadi pada rangkaian digital. Salah satu cara untuk mendeteksi kesalahan dengan menggunakan generator dengan sistem dan rangkaian digital yang realistis sehingga hasil yang ditemukan akurat dan cepat [5]. Generator jenis ini tersusun dari beberapa gerbang logika dan rangkaian digital. Rangkaian digital yang terdiri dari rangkaian logika kombinasional dengan beberapa rangkaian IC secara signifikan, berbeda dengan rangkaian digital regular yang hanya berisi logika saja [6]. Koneksi antara gerbang logika pada rangkaian IC akan berproses jika telah terdeteksi kesalahan di jalur masukannya. Penyimpangan pada pola keluarannya dapat dikelompokkan dalam himpunan kesalahan minimum [4].

Multi rangkaian terdiri dari beberapa sub-rangkaian kecil yang diekstraksi dan disintesis ulang menggunakan gerbang logika dua tingkat atau lebih yang ditujukan untuk meningkatkan keunggulannya [5]. Setiap sub-rangkaian yang telah diekstraksi dengan proses percepatan yang telah diterapkan guna menghasilkan sub-rangkaian multilevel. Aplikasi ini akan mengurangi area sub-rangkaian yang disintesis ulang tanpa berdampak negatif pada rangkaian logika tersebut. Teknik ini mengambil keuntungan dari varians probabilitas pola masukan dari sub-rangkaian yang telah diekstraksi, sehingga akan meningkatkan fleksibilitas dalam mendeteksi kesalahan pada rangkaian digital karena mempersingkat jalur rangkaian digital yang dapat dipertimbangkan untuk mendeteksi kemungkinan terjadinya kesalahan pada pola masukan rangkaian digital kombinasional [5].

Berdasarkan uraian tersebut, maka diciptakan sebuah rancangan generator set uji untuk diagnosis gangguan di rangkaian digital kombinasional. Selain itu generator set uji dibuat untuk mengikuti perkembangan teknologi elektronika yang berkembang pesat. Sehingga dapat memudahkan pendeteksian gangguan pada perangkat elektronika secara otomatis. Pada penelitian ini generator set uji dibangun sesuai dengan pohon diagnosa yang dihasilkan dari metode tabel kesalahan. Generator set uji mendiagnosis kesalahan pada rangkaian digital kombinasional dengan Arduino nano sebagai pengontrol utamanya. Generator

set tes memodelkan kesalahan yang menghasilkan perubahan nilai jalur rangkaian dari 0 ke 1, atau 1 ke 0. Masing-masing pengujian menggunakan kesalahan *stuck at 1* dan *stuck at 0*. Pengujian pada area rangkaian digital terjadi selama proses diagnosa kesalahan pada jalur masukan rangkaian digital kombinasional [7].

1.2 Rumusan Masalah

Mengacu pada permasalahan yang ada maka rumusan perancangan ini adalah sebagai berikut:

1. Bagaimana menentukan set pengujian minimal untuk diagnosis kesalahan pada suatu rangkaian digital kombinasional menggunakan metode tabel kesalahan?
2. Bagaimana cara membangun generator set tes untuk uji rangkaian digital kombinasional?

1.3 Tujuan Penelitian

Adapun tujuan yang ingin dicapai pada penelitian ini adalah sebagai berikut:

1. Membangun generator set uji untuk diagnosa gangguan pada rangkain *multiplexer quad 2-line to 1-line* IC 74157 menggunakan metode tabel kesalahan.
2. Menganalisa efisiensi pengujian berdasarkan pohon diagnosa yang didapatkan dari metode tabel kesalahan

1.4 Manfaat Penelitian

Melalui penelitian ini, manfaat yang dapat diperoleh dari sistem yang dibangun adalah kesalahan rangkaian digital, sehingga para perancang dapat mengetahui terjadinya gangguan lebih awal pada sistem digital kombinasional yang diciptakan. Sehingga dapat tercapai standar reabilitasnya dan dihasilkan produk sesuai dengan yang diharapkan dengan kualitas yang bagus.

1.5 Batasan Masalah

Adapun pembatasan masalah dalam penelitian ini adalah seperti yang di bawah ini:

1. Pendeteksian gangguan hanya pada kesalahan tunggal yang bersifat tetap atau permanen pada rangkaian digital kombinasional.
2. Objek pengujian menggunakan *multiplexer quad 2-line to 1-line IC 74157* IC 74157.
3. Generator set uji dibangun menggunakan Arduino nano atmega328.

1.6 Hipotesis

Metode tabel kesalahan menghasilkan set pengujian lengkap minimum yang dipetakan menjadi sebuah pohon diagnosa. Implementasi dari pohon diagnosa yaitu perancangan generator set uji yang berhasil mendeteksi kesalahan dan lokasi terjadinya kesalahan di setiap jalur *input* rangkaian *multiplexer quad 2-line to 1-line IC 74157*.

1.7 Sistematika Penulisan

Untuk memudahkan penulisan dan pemahaman mengenai materi penelitian ini, maka tulisan ini akan dibagi menjadi lima, yaitu:

BAB I. PENDAHULUAN

Pada bab ini memuat latar belakang perumusan masalah, tujuan penelitian, batasan masalah, manfaat penelitian, serta sistematika penulisan.

BAB II TINJUAN PUSTUKA

Pada bab ini membahas tentang teori-teori dasar yang mendukung penelitian ini, yang diambil dari berbagai sumber ilmiah yang digunakan dalam laporan penelitian ini.

BAB III METODE PENELITIAN

Pada bab ini menjelaskan waktu dan tempat penelitian, alat dan bahan yang digunakan dalam penelitian, metode penelitian yang digunakan, serta pelaksanaan dan pengamatan penelitian.

BAB IV HASIL DAN PEMBAHASAN

Pada bab ini menjabarkan data hasil penelitian dan pembahasan hasil data yang telah diperoleh dari penelitian ini.

BAB V KESIMPULAN

Pada bagian bab terakhir ini menjelaskan kesimpulan yang berdasarkan hasil data dan pembahasan yang mengacu pada tujuan penelitian.

II. TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

Ketika melaksanakan penelitian dibutuhkan peninjauan terkait dengan penelitian yang telah dilaksanakan atau penelitian sebelumnya supaya menghasilkan referensi yang sesuai dengan penelitian yang sedang dilakukan saat ini, yaitu tentang pendeteksian gangguan pada rangkaian digital kombinasional. Berikut beberapa uraian terkait dengan penelitian terdahulu yang digunakan sebagai acuan penelitian ini:

Berdasarkan penelitian Ageng Sadnowo Repelianto pada tahun 1996 dalam penelitiannya yang berjudul Penggunaan Metoda Tabel Kesalahan untuk Diagnosis Kesalahan Rangkaian Digital Kombinasional. Penelitian ini menggunakan metode tabel kesalahan untuk diagnosis kesalahan rangkaian kombinasional berdasarkan analisa kesalahan yang mungkin muncul pada rangkaian dan disusun dalam sebuah tabel kesalahan. Diharapkan panjang eksperimen deteksi tanpa kesalahan adalah 10 tingkat pengujian dari 4096 variasi pengujian dengan efisiensi pengujian 99,76%. Lokasi kesalahan pada modul menghasilkan panjang eksperimen deteksi tanpa kesalahan adalah 11 tingkat pengujian dengan efisiensi sebesar 99,73%. Pohon diagnosa mempercepat ditemukannya lokasi kesalahan yang terjadi pada rangkaian. Deteksi kesalahan tunggal memiliki rata-rata panjang pengujian 5,258 tingkat pengujian atau mempunyai efisiensi sebesar 47,42% dari panjang pengujian tanpa kesalahannya. Deteksi kesalahan pada modul mempunyai rata-rata panjang pengujian adalah 5,35 tingkat pengujian atau mempunyai efisiensi sebesar 51,36% dari panjang pengujian tanpa kesalahannya [4].

Berdasarkan penelitian Akshay Baid dkk pada tahun 2013 yang berjudul *Generating Test Patterns for Fault Detection in Combinational Circuits Using Genetic Algorithm*. Penelitian ini bertujuan untuk pendeteksian kesalahan otomatis dan uji minimalisasi berdasarkan model kesalahan *stuck-at* pada produksi *chip* sebelum dipasarkan dan digunakan secara massal menggunakan algoritma genetika. Pengujian ini dicapai dengan menghasilkan pola pengujian dalam cakupan kesalahan yang tinggi serta harus menjadikan rangkaian pengujian minimal untuk mengurangi biaya produksi dengan cara menghemat waktu produksinya. Cacat pada *chip* diasumsikan sebagai kesalahan tetap atau permanen yang bersifat tidak sementara, artinya tanpa diperbaiki atau diperbaiki kesalahan tersebut akan tetap ada. Rangkaian yang digunakan adalah rangkaian logika resistor transistor (RTL), rangkaian logika diode transistor (DTL) dan rangkaian logika transistor-transistor (TTL). Proses pengujian, menggunakan model kesalahan tunggal *stuck-at*, semua gangguan *stuck-at 0* dan *stuck-at 1* dalam sebuah rangkaian dapat dideteksi jika rangkaian tersebut tidak redundan serta fungsi yang direalisasikan adalah fungsi yang diminimalisasikan. Kesalahan tidak dapat dibedakan jika tabel kebenaran dari fungsi keluaran rangkaiannya adalah identik. Penerapan dilakukan dengan populasi awal 10 kromosom untuk rangkaian yang lebih kecil dan 25 kromosom untuk rangkaian yang lebih besar dihasilkan secara acak. Kemudian kesalahan disimulasikan dan dilihat apakah keluaran yang diperoleh berbeda dari respon bebas kesalahan. Algoritma genetika membantu menemukan dengan cepat kumpulan pola pengujian optimal yang dapat mendeteksi kesalahan pada sebuah rangkaian. Algoritma ini dapat digunakan dalam generator pola pengujian otomatis untuk mendeteksi kesalahan yang diberikan dengan model *stuck-at*. Perbaikan lebih lanjut dapat dilakukan dengan menyertakan kesalahan tunda dan mengimplementasikan untuk rangkaian berurutan. Hasilnya untuk rangkaian *benchmark ISCAS 1989* pada rangkaian C17 yang diberikan satu kesalahan yaitu pada jalur G10 *stuck-at 0* didapatkan 6 iterasi cakupan kesalahan dari 100 cakupan kesalahan [8].

Berdasarkan penelitian Yu Zhang dkk pada tahun 2014 yang berjudul *Diagnostic Test Generation for Transition Delay Faults Using Stuck-At Fault*

Detection Tools. Penelitian ini bertujuan untuk meningkatkan kepadatan logika, kecepatan, dan waktu terhadap tekanan *chip VLSI modern*. Diagnosis kesalahan merupakan langkah yang penting dalam mengisolasi lokasi jaringan cacat produksi yang paling mungkin terjadi sehingga identifikasi dan eliminasi cacat produksi dapat dilakukan. *Delay* diagnosis kesalahan merupakan karakteristik penting kinerja perangkat *VLSI modern* dan cakupan diagnostik yang tinggi dari pengujian yang diinginkan. Menambahkan beberapa gerbang logika dengan satu atau dua model *flip-flop* ke rangkaian yang diuji. Membuat model deteksi atau generator pola diagnosis uji otomatis dari model kesalahan tunda transisi yang dapat digunakan oleh generator pola uji kesalahan tunggal *stuck-at* konvensional. Metode yang digunakan yaitu kesalahan penundaan transisi. Model generator pola diagnosis uji otomatis dapat diperluas menjadi dua kerangka waktu untuk memfasilitasi penggunaan generator pola diagnosis uji otomatis kombinasional. Waktu yang dihasilkan pada rangkaian yang diuji adalah 14841 detik untuk 1645 percobaan eksklusif. Percobaan ini menemukan bahwa hampir 99,8% waktu dihabiskan dalam rekonstruksi struktur data. Waktu yang dibutuhkan untuk proses pembuatan uji diagnostik akan berkurang menjadi beberapa menit untuk rangkaian besar dan detik untuk rangkaian kecil. Akibatnya, presentase pasangan kesalahan penundaan transisi yang berbeda lebih besar dan sistem pembangkitan uji eksklusif otomatis yang diusulkan lebih efisien waktu [9].

Berdasarkan penelitian Ashwani Kumar dkk pada tahun 2016 yang berjudul *Transistor Level Fault Diagnosis in Digital Circuits Using Artificial Neural Network*. Jaringan syaraf tiruan dalam penelitian ini diaplikasikan untuk diagnosis kesalahan pada rangkaian digital, gangguan level transistor terjadi karena korsleting di terminal transistor dengan variasi parameter transistor. Selama proses diagnosis, variasi parametrik dalam transistor juga diperhitungkan dengan memvariasikan tegangan ambang transistor. Jaringan syaraf tiruan yang digunakan dalam klasifikasi kesalahan pada gerbang NAND menggunakan 25 neuron lapisan tersembunyi. Terdapat 462 periode jaringan syaraf tiruan yang dilatih. Jaringan syaraf tiruan yang digunakan untuk klasifikasi kesalahan pada rangkaian gerbang NOR menggunakan 30 *neuron*

lapisan tersembunyi dan dilatih dalam 452 periode. Semua model kesalahan yang ditentukan terdiagnosis dengan benar, menghasilkan efisiensi diagnosis yang sama seperti yang diperoleh pada rangkaian gerbang NAND [10].

Berdasarkan penelitian Soham Roy dkk pada tahun 2021 dengan judul *Training Neural Network for Machine Intelligence in Automatic Test Pattern Generator*. Penelitian ini menunjukkan bahwa jaringan syaraf tiruan dapat menggabungkan beberapa *heuristic* untuk memandu generator pola pengujian otomatis dengan *backtrack* lebih sedikit daripada yang dibutuhkan oleh panduan dari *heuristic* tunggal manapun. Metode yang digunakan yaitu dengan mengembangkan metode pelatihan baru untuk memasukan beberapa heuristik. Jaringan syaraf tiruan memiliki neuron keluaran tunggal dan satu lapisan neuron tersembunyi, yang cukup untuk mengkomodasi volume data pelatihan. Generator pola pengujian otomatis konvensional diterapkan pada kesalahan yang sulit dideteksi dan mudah dideteksi pada rangkaian *benchmark* yang dipilih memberikan data pelatihan untuk *node* yang ditandai sebagai “sukses”. Jika *backtrace* mengarah ke pengujian dan “gagal” jika menghasilkan *backtrack*. Eksperimen menggunakan semua kesalahan untuk menunjukkan keakuratan jaringan syaraf tiruan menggunakan teknik pelatihan yang diusulkan. Pengamatan penting yang dilakukan yaitu pada C17, B02, dan B01 tidak memiliki *fanout konvergen* sehingga tidak memiliki *backtrack*. Tidak ada ruang untuk mengurangi *backtrak* dengan panduan JST baru. Jumlah *backtrace* adalah konstan atau berkurang dalam rangkaian bebas *fanout rekonvergen* ini oleh JST baru kecuali pada C1908, C432, C499, dan B05. Berdasarkan tabel data hasil yang menunjukkan waktu komputasi generator pola uji otomatis “CPU Time (ms)”, “*Backtrace Count*”, dan “*Backtrack Count*”. Jaringan syaraf tiruan baru bekerja lebih baik dengan *backtracks* lebih sedikit dari jaringan syaraf tiruan aslinya [5].

Berdasarkan penelitian Isha Gupta pada tahun 2022 yang berjudul *Stuck at Fault Testing in Combinational Circuit Using FPGA*. Pendekatan berbasis FPGA untuk generator pola pengujian dan pengujian rangkaian kombinasional. Konsep emulasi disajikan untuk mendeteksi kesalahan dalam rangkaian

menggunakan perangkat keras FPGA. Permasalahan sedikitnya *port* masukan dan keluaran yang tersedia pada papan FPGA telah diatasi melalui pendekatan *multiplexing* sakelar. Papan FPGA yang digunakan pada penelitian ini adalah papan pengembangan berbasis *Spartan 6* FPGA yang dikembangkan oleh CDAC Noida. Modul rangkaian *benchmark C17* tanpa kesalahan dikodekan di *Verilog*. Kemudian dua Salinan tambahan dari modul C17 yang sama dibuat *multiplexer* dan dimasukkan pemeriksaan untuk injeksi kesalahan berdasarkan jalur yang dipilih. Hal yang sama diterapkan pada papan FPGA berbasis *Spartan 6*, masukan diberikan melalui sakelar dan tombol. Terdapat 12 sakelar akan dipetakan ke 12 masukan dan proses akan diulangi tergantung dengan jumlah total masukan yang akan diberikan. Dengan demikian *finite state machine* diimplementasikan untuk memetakan perilaku transisi, kondisi ini yang dapat mengetahui berapa kali tombol telah ditekan. Setelah semua masukan telah diberikan, keluaran dieksekusi dan kesalahan dapat dideteksi. Keluaran diteruskan melalui LED pada papan FPGA. Keluarannya adalah EXOR dari keluaran yang diperoleh pada rangkaian bebas kesalahan dan rangkaian yang rusak. Setiap kali LED menyala, menggambarkan bahwa keluaran yang diperoleh berbeda, ini menyiratkan bahwa kesalahan telah mengubah hasil keluaran. Pengujian ini menghasilkan persentase pemanfaatan untuk irisan register, LUT dan angka sebagai logika sebesar 1%,3%, dan 3%. Persentase jumlas irisan dan jumlah MUXCY yang digunakan adalah 4% [11].

Secara garis besar, penelitian-penelitian terdahulu memiliki kelebihan dan kekurangannya masing-masing. Berdasarkan penelitian ini melalui metode yang telah ditentukan diharapkan dapat memberikan informasi penting untuk kemajuan dunia industri digital dan mikrokomponen. Keuntungan yang ditawarkan pada penelitian ini adalah penggunaan komponen dan metode yang lebih sederhana sehingga lebih efisien dan cepat dalam mendeteksi kesalahan pada rangkaian digital kombinasional.

2.2 Gangguan Digital

Gangguan merupakan sebuah kerusakan yang berbentuk fisik. Cacat yang biasa terjadi pada komponen-komponen dalam peralatan elektronika, baik pada perangkat lunak maupun perangkat keras. Gangguan sendiri bisa berbentuk sebuah kesalahan, seperti penyimpangan yang terjadi dari hasil sebenarnya pada sebuah fungsi rangkaian. Berdasarkan perancangan toleransi gangguan, gangguan terdiri dari tiga bentuk dasar yaitu, gangguan, kesalahan, dan kegagalan. Ketiga bagian tersebut memiliki hubungan yang saling berkaitan [6]. Gangguan merupakan penyebab terjadinya kesalahan serta kesalahan merupakan penyebab dari terjadinya kegagalan suatu sistem rangkaian elektronika [2].

Apabila hasil dari suatu kesalahan yang memiliki fungsi tidak sesuai dalam sebuah sistem, maka akan terjadi sebuah kegagalan pada sistem tersebut [12]. Sehingga kegagalan merupakan perbuatan yang bukan sifat sebenarnya dari sebuah sistem, seperti pada Gambar 2.1.



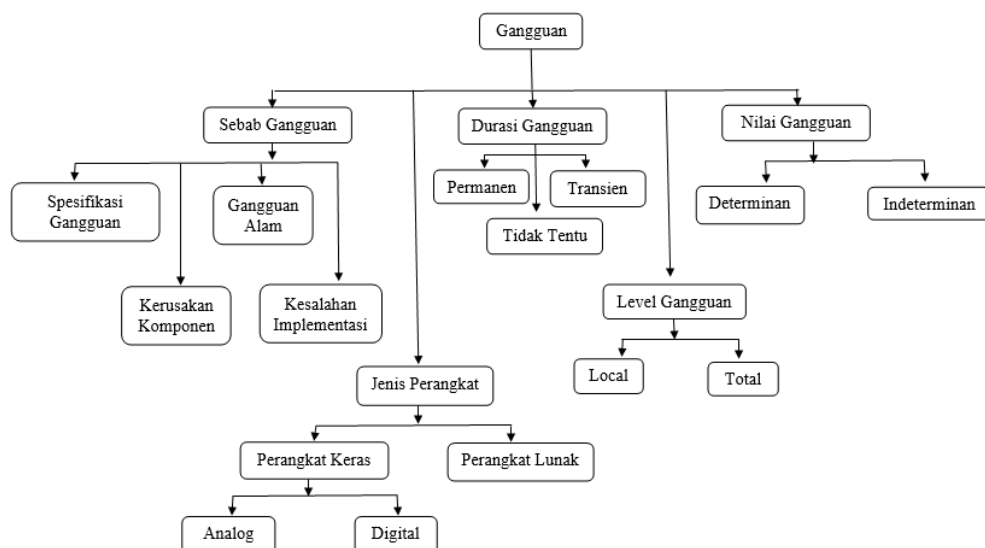
Gambar 2.1 Ikatan antara Gangguan, Kesalahan, dan Kegagalan

2.3 Karakteristik Gangguan

Karakteristik gangguan terdiri dari beberapa aspek, salah satunya yaitu dari segi durasi atau lamanya gangguan terjadi. Gangguan permanen merupakan karakteristik gangguan yang tidak akan hilang apabila tidak ditangani dengan cepat, akibat yang akan ditimbulkan yaitu sistem kerja dari rangkaian logika tidak bisa bekerja dengan baik, sehingga perangkat elektronika tersebut rusak dan tidak akan bisa digunakan [6]. Adapun contoh dari jenis karakteristik gangguan permanen, yaitu terjadinya kerusakan jalur atau gerbang logika yang

disebabkan oleh kerusakan fisik komponen pada perangkat elektronika atau karena dari jalur gerbang logika perangkat itu sendiri.

Gangguan transien merupakan karakteristik gangguan yang timbul dan dapat menghilang dalam jeda waktu tertentu [13]. Salah satu contoh gangguan transien yaitu gangguan yang disebabkan dari faktor luar seperti halilintar. Halilintar dapat menyebabkan gangguan sebuah sistem dalam jangka waktu tertentu, namun tidak merusaknya. Setelah halilintar menghilang maka gangguan transien juga menghilang [3]. Gangguan tidak menentu merupakan salah satu gangguan yang bisa muncul, menghilang dan kemudian muncul kembali secara berulang-ulang. Jenis karakteristik gangguan seperti Gambar 2.2.



Gambar 2.2 Karakteristik Gangguan

2.4 Gerbang Logika

Gerbang-gerbang logika utama merupakan sebuah elemen dasar yang dibentuk menjadi sebuah rangkaian digital yang mempunyai kesatuan dari beberapa gerbang-gerbang logika dasar yang bentuknya berfungsi untuk memproses sinyal digital [14]. Gerbang logika umumnya terdiri atas tiga gerbang utama diantaranya, gerbang NOT, Gerbang AND dan gerbang OR [1]. Adapun

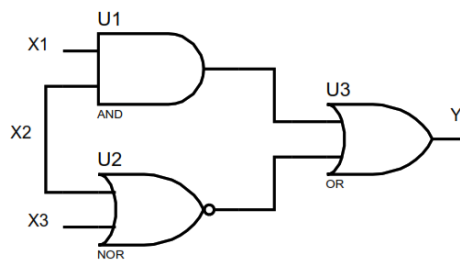
gerbang lain selain gerbang utama yaitu seperti gerbang NAND, gerbang NOR, gerbang XOR dan gerbang XNOR merupakan kombinasi dari tiga gerbang logika utama tersebut [4].

2.5 Rangkaian Kombinasional

Rangkaian gabungan atau yang biasa disebut dengan rangkaian kombinasional merupakan sebuah rangkaian yang memiliki besaran keluaranya (*output*) bergantung pada besaran masukannya (*input*) [15]. Prinsip kerja dari rangkaian kombinasional memiliki nilai pada keluaranya yang ditetapkan berdasarkan besaran masukannya pada waktu tertentu, hal ini dikarenakan rangkaian kombinasional tidak mempunyai karakter memori. Beberapa contoh pengaplikasian dari rangkaian digital kombinasional diantaranya, *adder*, komperator, *Demultiplexer*, *multiplexer*, *decoder* dan *encoder* [1]. Rangkaian digital kombinasional yang iredundan terjadi akibat kesalahan logika yang terjadi pada setiap bagian dari rangkaian yang akan menyebabkan perubahan dalam fungsi penyambungan yang diwujudkan oleh rangkaian bebas kesalahan tersebut [9].

Proses kerja rangkaian digital kombinasional menggunakan prinsip aljabar Boolean. Aljabar Boolean merupakan perpaduan dari salah satu aljabar matematika yang mempunyai nilai kebenaran yaitu *true* dan *false* nilai kesalahan [14]. Nilai kebenaran dilambangkan dengan angka 1 dan nilai kesalahan dilambangkan dengan angka 0. Boolean disebut juga dengan aljabar biner yang hanya mengenal dua masukan yaitu 0 dan 1. Aljabar Boolean dikenal juga untuk digunakan dalam menganalisa dan menyederhanakan rangkaian digital atau gerbang logika [15]. Berdasarkan Gambar 2.4 didapatkan persamaan Boolean 2.1.

$$Y = f(X_1, X_2, X_3) = X_1 X_2 + (\overline{X_2} + \overline{X_3}) \quad (2.1)$$



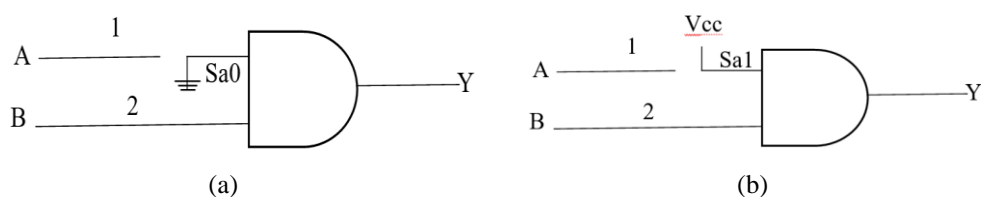
Gambar 2.3 Rangkaian Digital Kombinasional

2.6 Diagnosa Kesalahan

Diagnosa kesalahan merupakan proses pendeteksian kesalahan serta mencari letak terjadinya kesalahan. Mendiagnosa kesalahan dibutuhkan sebuah set pengujian, yaitu sebuah set minimal yang dapat mendeteksi masing-masing kesalahan yang sedang diteliti [4]. Penyusunan set pengujian deteksi kesalahan minimal untuk rangkaian digital kombinasional dibutuhkan set pengujian lengkap dari setiap kesalahan yang sedang diteliti [8]. Sebuah pengujian kesalahan dapat diartikan sebagai aplikasi dari sebuah pola masukan pada masukan utama dan pemantau pada keluaran utama sehingga adanya kesalahan bisa dideteksi dengan sebuah set pengujian lengkap dari sebuah kesalahan yang berisi semua pengujian kesalahan.

2.7 Model Gangguan *Stuck Logika*

Penerapan model gangguan *stuck* logika sangat bervariasi dan sangat luas karena model ini sangat sederhana dan efektif penggunaannya. Model gangguan ini bisa sebagai gangguan *stuck at 0* (Sa0) atau sebagai gangguan *stuck at 1* (Sa1) [4].



Gambar 2.4 Konsep dasar Model Gangguan *Stuck Logika* (a) *Stuck at 0*, (b) *Stuck at 1*

Secara sederhana gangguan pada suatu rangkaian digital dimodelkan menjadi kesalahan logika *stuck*, masukan gerbang mengalami *stuck at 1* (Sa1) atau *stuck at 0* (Sa0) yang analoginya bisa dilihat pada Gambar 2.3. Sa1 adalah kondisi seolah-olah *input* gerbang mengalami situasi selalu terhubung logika 1 meskipun *input* A bernilai 1 ataupun 0. Sementara itu, Sa0 kondisi dimana *input* gerbang terkondisi pada logika 0, meskipun *input* a bernilai 1 atau 0 [17].

Tabel 2.1 Keluaran Gerbang *AND* saat Normal dan Dalam Kondisi *Stuck*

Uji	B	A	y_0	y_{10}	y_{11}	y_{20}	y_{21}
0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	1
2	1	0	0	0	1	0	0
3	1	1	1	0	1	0	1

Tabel 2.1 menunjukkan kondisi gerbang *AND* dalam kondisi normal y_0 , dan kondisi jalur *input* A dan B mengalami gangguan Sa0 dan Sa1 berturut-turut y_{10} , y_{11} , y_{20} dan y_{21} . Pola *output* yang sama terlihat pada y_{10} dan y_{20} , kedua pola tidak dapat dibedakan oleh set uji [17]. Dengan demikian kedua kondisi dikelompokkan dalam satu pola kesalahan f_1 seperti pada tabel 2. Secara lengkap terdapat 4 kelompok kesalahan. f_0 adalah pola *output* tanpa kesalahan. Kemudian f_1 adalah pola *output* yang menunjukkan adanya gangguan di lokasi jalur 1 atau jalur 2 mengalami Sa0. f_2 adalah pola *output* dengan gangguan di lokasi jalur 1 mengalami Sa1. Terakhir f_3 adalah pola *output* yang menunjukkan gangguan di lokasi jalur 2 dengan Sa1 [4].

Tabel 2.2 Lokasi Kesalahan Gerbang *AND* Saat Normal dan Dalam Kondisi *Stuck*.

Uji	B	A	$f_0 = \{y_0\}$	$f_1 = \{y_{10}, y_{20}\}$	$f_2 = \{y_{11}\}$	$f_3 = \{y_{21}\}$
0	0	0	0	0	0	0
1	0	1	0	0	0	1
2	1	0	0	0	1	0
3	1	1	1	0	1	1

Pada Tabel 2.2 pengujian (0) adalah redundan yang tidak dapat membedakan kesalahan karena tidak ada hasil ketika menggunakan operasi X-OR, sehingga pengujian {1, 2, 3} adalah set pengujian lengkap minimal untuk gerbang *AND*.

Tabel 2.3 Pemilihan Set Pengujian Lengkap

Uji	X-OR Set Kesalahan									
	f	0	0	0	f	1	1	f	2	3
0										
1				1		1			1	
2			1			1				
3		1				1	1			

Keterangan: $f_{ab} = f_a \text{ X-OR } f_b$

2.8 Pohon Diagnosa

Pohon diagnosa merupakan sebuah grafik penunjuk yang mempunyai simpul berupa pengujian. Cabang yang dihasilkan dari simpul-simpul menyatakan akibat yang berbeda dari pengujian khusus. Masing-masing cabang menyatakan kesalahan-kesalahan yang mungkin ada didalam rangkaian tersebut. akar cabang dari pohon diagnosa memberikan petunjuk apakah rangkaian bebas dari kesalahan atau ada kemungkinan terjadinya kesalahan [4]. Selanjutnya, menentukan urutan variabel uji untuk membentuk pohon diagnosa. Langkah pertama, menentukan selisih jumlah 0 dan 1 dalam setiap baris. Tujuan R_i untuk menyeimbangkan jumlah 0 dan 1 supaya cepat dicarinya sehingga selisih yang paling kecil dipilih sebagai uji yang terdahulu. Untuk mencari R_i menggunakan persamaan 2.2 sebagai berikut:

$$R_i = |w_{i0} - w_{i1}| \quad (2.2)$$

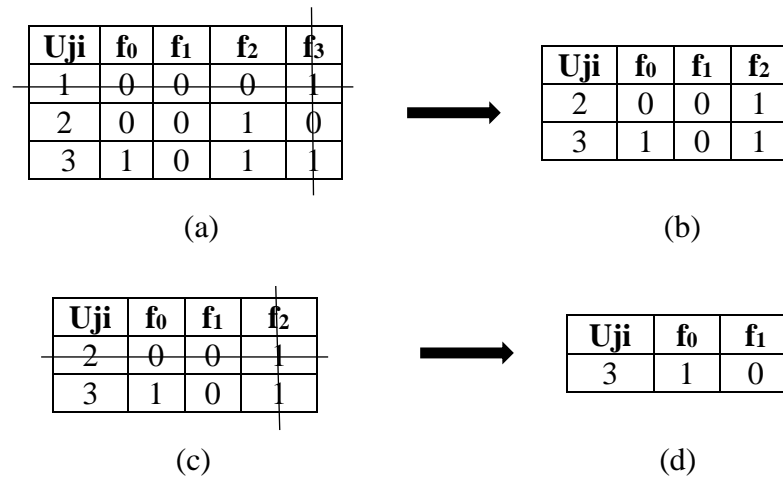
Dimana: R_i merupakan selisih terkecil antara jumlah 0 dan 1

w_{i0} adalah jumlah dari 0

w_{i1} adalah jumlah dari 1

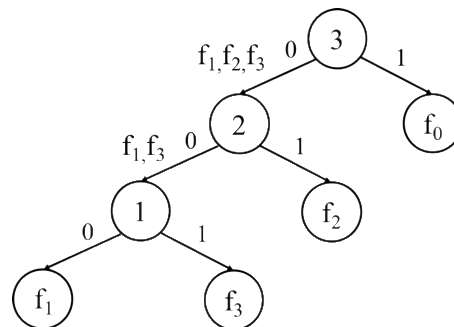
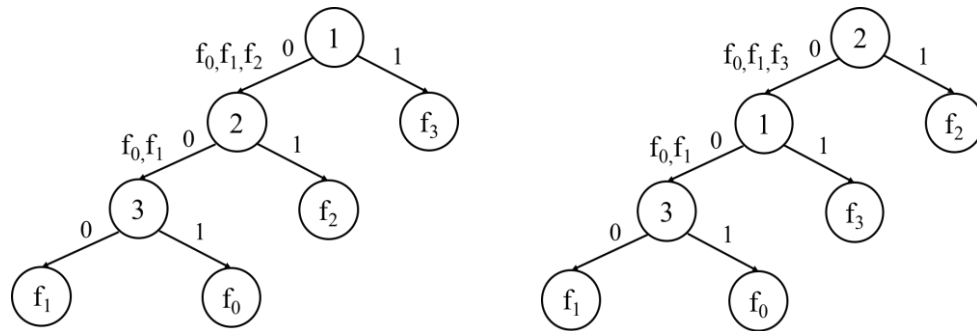
Gambar 2.5 merupakan baris uji 1, 2 dan 3 yang memiliki selisih jumlah 0 dan 1 yang sama, yaitu 2. Oleh karenanya, sebagai variabel uji pertama dapat dipilih 1, 2 atau 3. Jika uji 1 yang dipilih, dan *output* menghasilkan 1, pengujian sudah dapat mengenali bahwa gerbang mengalami gangguan di lokasi dengan definisi f_3 . Namun jika *outputnya* 0, hal ini mengindikasikan gerbang dalam kondisi f_0 ,

f_1 , atau f_2 . Kemudian, usaha yang sama dilakukan agar f_0 , f_1 , atau f_2 dapat dibedakan.



Gambar 2.5 Menentukan Urutan Set Uji Pohon Diagnosa

Langkah kedua mengecilkan matrik pencarian dengan menghilangkan komponen bernilai 1 yang dalam kekuasaan kolom terhadap baris seperti pada tabel 3(a) dan membentuk tabel 3(b). Terdapat komponen uji 2 dan 3. Pilih salah satu diantaranya dengan melakukan langkah pertama. Kedua komponen uji memiliki selisih jumlah 0 dan 1 yang sama, karenanya, dapat dipilih uji 2 atau uji 3 sebagai set uji ke dua. Jika uji 2 yang dipilih keluaran gerbang AND yang bernilai 1 akan mendeteksi adanya gangguan di lokasi dengan kondisi kesalahan f_2 . Selanjutnya Pengujian dengan set uji 3, akan membedakan kondisi gerbang dalam kondisi baik f_0 jika *output* gerbang bernilai 1, atau mengindikasikan gerbang dalam gangguan dengan kondisi f_1 [5]. Urutan pengujian ini digambarkan dalam sebuah pohon diagnosa bisa dilihat pada Gambar 2.6(a). Gambar 2.6(b) dan 2.6(c) adalah alternatif prioritas set uji.



Gambar 2.6 Pohon Diagnosa Gerbang AND

Hasil dari pohon diagnosa pada Gambar 2.5 diimplementasikan menjadi sebuah generator set uji [17]. Efisiensi yang dihasilkan oleh generator set uji didapatkan dengan persamaan sebagai berikut [4]:

$$Efisiensi (\%) = \frac{2^n - \text{Level Pengujian Set Kesalahan}}{2^n} \times 100 \quad (2.3)$$

Dimana: n adalah jumlah *input* rangkaian uji

III. METODE PENELITIAN

3.1 Waktu dan Tempat Penelitian

Penelitian dan perancangan tugas akhir ini dilaksanakan dari bulan Agustus 2022 sampai Juni 2023, bertempat di Laboratorium Terpadu Teknik Elektro, Jurusan Teknik Elektro, Universitas Lampung.

3.2 Alat dan Bahan Penelitian

Adapun alat dan bahan pada penelitian tentang rancang bangun generator tes ini adalah sebagai berikut:

1. 1 Unit Laptop Asus
2. *Software* GUI
3. 8 Buah *Project Board*
4. 1 Buah Arduino Nano
5. 1 Buah Ic 74157
6. 4 Buah Ic 7411 (*AND*)
7. 4 Buah Ic 7432 (*OR*)
8. 4 Buah Ic 7414 (*NOT*)
9. 88 Buah Sakelar SPDT
10. 1 Buah Multimeter

3.3 Prosedur Penelitian

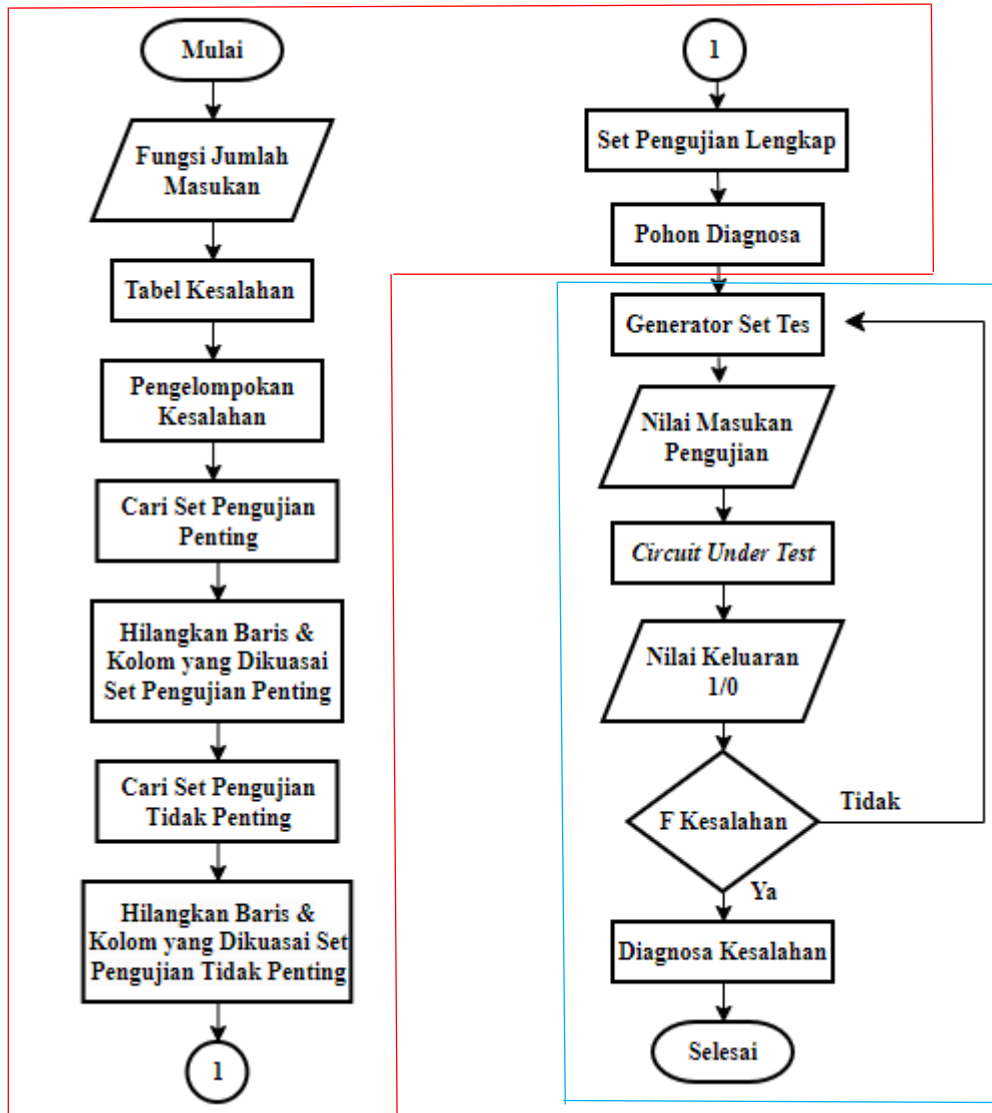
Metode penelitian yang digunakan dalam penelitian ini *Research and Development*. (R&D). Penelitian serta pengembangan merupakan metode penelitian yang digunakan untuk menghasilkan serta mengembangkan suatu produk dan menguji validitas produk yang dihasilkan.

Pada penelitian ini didapatkan set pengujian lengkap menggunakan metode tabel kesalahan yang dipetakan menjadi pohon diagnosa. Pohon diagnosa diimplementasikan menjadi generator set uji yang menghasilkan efisiensi waktu dalam mendiagnosis kesalahan pada rangkaian kombinasional.

3.4 Metode yang Digunakan

Secara keseluruhan tahapan penelitian dibuat dalam bentuk diagram alir dan diagram tulang ikan. Diagram alir perancangan generator set uji tersusun dari dua bagian utama. Bagian pertama yang ditandai dalam garis merah adalah penurunan set uji yang dipetakan pada pohon diagnosa. Bagian kedua yang ditandai dalam garis biru adalah implementasi pohon diagnosa menjadi sebuah generator set uji. Perancangan tersebut bisa dilihat pada Gambar 3.1.

Diagram tulang ikan menjelaskan tahapan yang dilakukan dengan rinci pada penelitian ini, dapat dilihat pada Gambar 3.2.



Gambar 3.1 Diagram Alir Perancangan Generator Set Uji

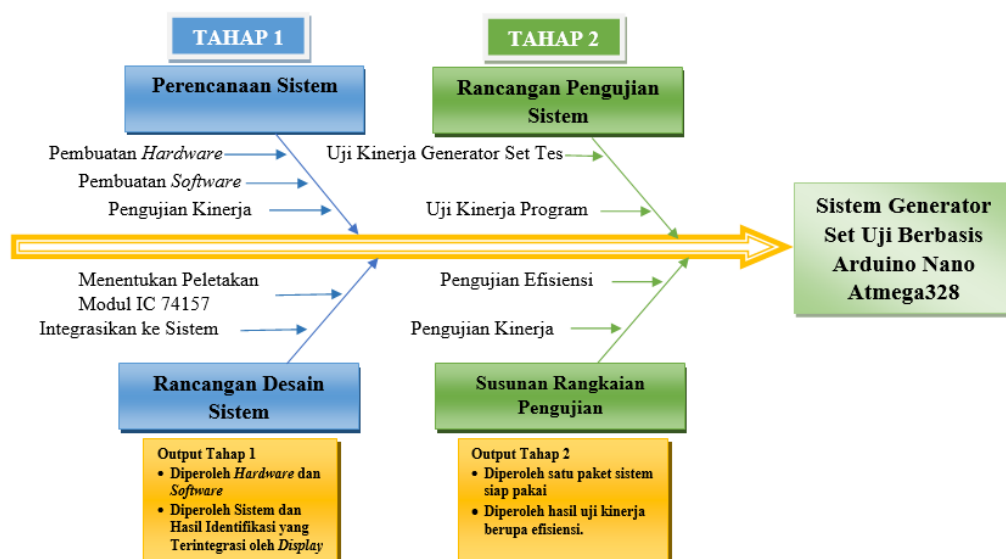
Tahap 1

Perencanaan sistem, pada proses ini dilaksanakan perancangan *hardware* dan *software* lalu melakukan pengujian kinerja. Sehingga diperoleh sistem yang diinginkan. Komponen sistem yang dirancang terdiri dari Arduino Nano Atmega328, IC 7408 (gerbang *AND*), IC 7432 (gerbang *OR*), IC 7414 (gerbang *NOT*) dan saklar SPDT 3 posisi.

Perancangan *display* hasil identifikasi, pada proses ini dilaksanakan perancangan skematik, *display* diintegrasikan dengan sistem lalu melakukan pengujian kinerja sistem. Hasil diagnosis kesalahan pada sistem generator set tes akan ditampilkan menggunakan *software* GUI melalui layar laptop. Pada tahap ini didapatkan hasil yaitu berupa rancangan *hardware* dan *software* yang terintegrasi dengan sistem dan *display* hasil diagnosa kesalahan generator set tes menggunakan *software* GUI.

Tahap 2

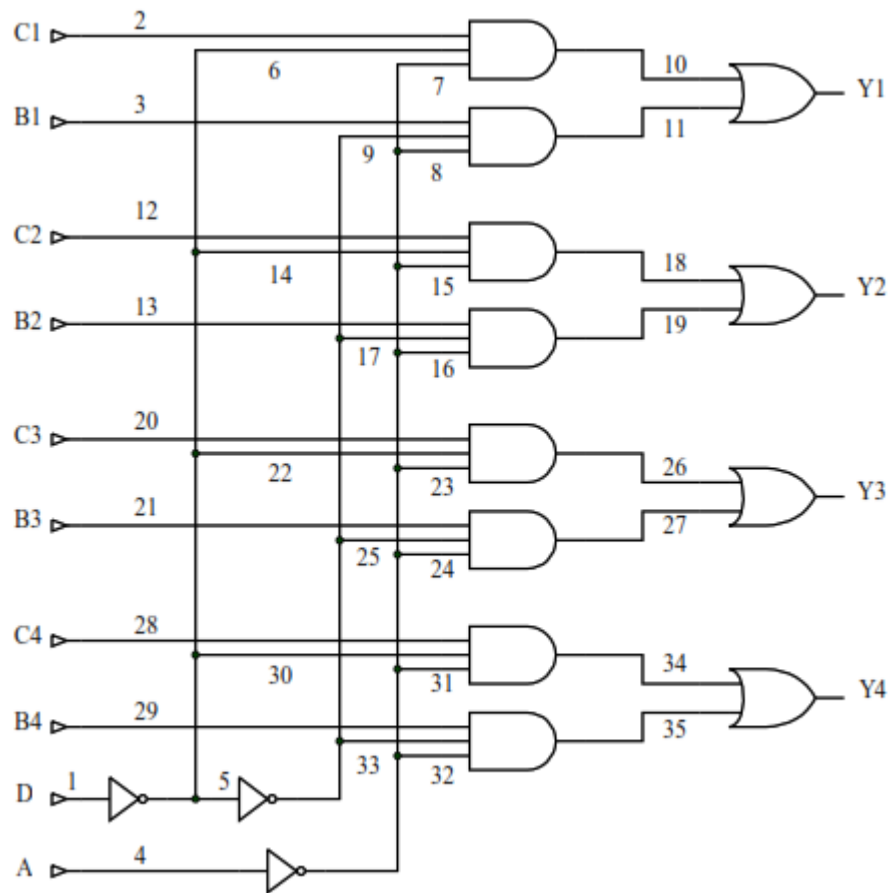
Pengujian kinerja serta akurasi sistem secara keseluruhan, proses ini adalah proses akhir pada perancangan sistem, pengujian kinerja yaitu akurasi dan stabilitas. Pengujian dilakukan dengan cara menggunakan susunan modul-modul IC menjadi rangkaian *multiplexer quad 2 line to 1 line* IC 74157 dalam kondisi nyata. Desain pengujian sama dengan pengujian lapangan awal dengan menggunakan metode eksperimen. Bila pengujian generator set tes belum memenuhi spesifikasi sesuai dengan yang diinginkan, maka perlu ada revisi terhadap produk tersebut. Hasil revisi selanjutnya digunakan untuk pengujian lapangan operasional. Pengujian lapangan operasional dilakukan dengan cara menguji sistem pengontrolan diantaranya identifikasi dan karakterisasi IC 74157. Tahapan ini diharapkan menghasilkan data akurat dari sistem pengontrol generator set tes.



Gambar 3.2 Diagram Tulang Ikan Tahapan Penelitian

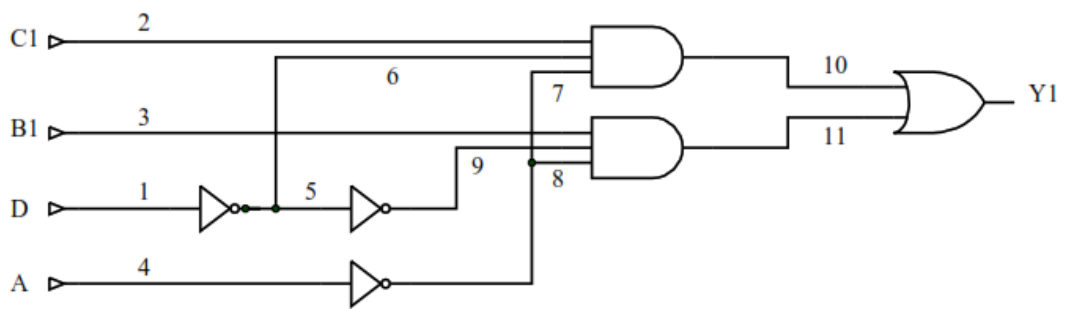
3.5 Susunan Rangkaian Pengujian

Sebagai aplikasi yang akan dicontohkan penyelesaian untuk penurunan eksperimen pengujian kesalahan dengan menggunakan rangkaian *multiplexer quad 2 line to 1 line* IC 74157. Rangkaian ini terdiri dari 15 gerbang logika yaitu tiga gerbang *NOT*, delapan gerbang *AND* dan empat gerbang *OR* dan 35 jalur masukan.



Gambar 3.3 Rangkaian *Multiplexer Quad 2 Line to 1 Line* IC 74157

Untuk mempermudah penurunan pengujian dilakukan penamaan untuk semua gerbang masukan dan jalur keluaran.



Gambar 3.4 Penyederhanaan Rangkaian untuk Penurunan Pengujian

Dalam menyusun tabel kesalahan untuk deteksi semua kesalahan tunggal dari rangkaian kombinasional perlu diperhatikan fungsi keluarannya seperti persamaan Boolean 2.1. Fungsi keluaran dari rangkaian Gambar 3.4, yaitu:

$$Y_1 = \bar{A}C1\bar{D} + \bar{A}B1D \quad (3.1)$$

Rangkaian multiplexer IC 74157 dapat disederhanakan menjadi empat potongan rangkaian uji karena memiliki pola yang sama. Persamaan boolean 3.1 dihasilkan dari potongan rangkaian uji pertama seperti Gambar 3.4.

Jika rangkaian IC 74157 Gambar 3.3 dipotong seperti Gambar 3.4 maka di hasilkan persamaan untuk potongan ke-2, ke-3 dan ke-4 dari rangkaian multiplexer IC 74157 adalah sebagai berikut:

$$Y_2 = \bar{A}C2\bar{D} + \bar{A}B2D \quad (3.2)$$

$$Y_3 = \bar{A}C3\bar{D} + \bar{A}B3D \quad (3.3)$$

$$Y_4 = \bar{A}C4\bar{D} + \bar{A}B4D \quad (3.4)$$

Persamaan di atas didapatkan karena input yaitu B2, C2 dengan keluaran Y₂ dan B3, C3 dengan keluaran Y₃ serta B4 dan C4 dengan keluaran Y₄ memiliki pola yang identik dengan masukan B1, C1 dengan keluaran Y₁.

Tabel 3.1 Tabel Kebenaran IC 74157

No	Masukan				Keluaran y
	D	C	B	A	
1	X	X	X	H	L
2	L	L	X	L	L
3	L	H	X	L	H
4	H	X	L	L	L
5	H	X	H	L	H

Keterangan: *L* = Low, *H* = High, dan *X* = Tidak ada

3.6 Metode Tabel Kesalahan

Metode tabel kesalahan adalah metode untuk mendeteksi dan memetakan kesalahan logika permanen atau kesalahan tunggal dalam suatu rangkaian digital kombinasional. Metode tabel kesalahan menurunkan set pengujian yang didapatkan dengan membandingkan tabel kebenaran dari rangkaian normal dan rangkaian yang mengalami kesalahan. Dalam penurunan panjang sebuah set pengujian, penjadwalan pengujian tidak bergantung pada hasil urutan pengujian tersebut.

3.6.1 Penyusunan Tabel Kesalahan

Gangguan pada rangkaian Gambar 3.4 dimodelkan dengan kesalahan *logic stuck* pada setiap jalur masukannya. Masukan gerbang mengalami *stuck at 1* (*Sa1*) atau *stuck at 0* (*Sa0*). Rangkaian Gambar 3.4 dalam kondisi normal atau tanpa adanya kesalahan yaitu y_0 , serta kondisi jalur *input* A, B1, C1 dan D mengalami gangguan *Sa0* dan *Sa1* berturut-turut pada potongan rangkaian pertama IC 74157. Hasil keluaran potongan rangkaian pertama didapatkan menggunakan Persamaan 3.1 yang bisa dilihat pada Tabel 3.2.

Tabel 3.2 Tabel Kesalahan Diagnosa Semua Kesalahan Tunggal dari Rangkaian 1

U	Masukan				Nilai Keluaran																							
	D	C	B1	A1	y ₀	y ₁₀	y ₁₁	y ₂₀	y ₂₁	y ₃₀	y ₃₁	y ₄₀	y ₄₁	y ₅₀	y ₅₁	y ₆₀	y ₆₁	y ₇₀	y ₇₁	y ₈₀	y ₈₁	y ₉₀	y ₉₁	y ₁₀₀	y ₁₀₁	y ₁₁₀	y ₁₁₁	
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
2	0	0	1	0	0	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	0	1	0	1
3	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
4	0	1	0	0	1	1	0	0	1	1	1	1	0	0	1	0	1	0	1	0	1	1	1	0	1	1	1	1
5	0	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	1	0	1	
6	0	1	1	0	1	1	1	0	1	1	1	1	0	1	1	1	1	0	1	0	1	1	1	0	1	1	1	
7	0	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	1	0	1	
8	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
9	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
10	1	0	1	0	1	0	1	1	1	0	1	1	0	1	0	1	0	0	1	0	1	0	1	1	1	1	0	1
11	1	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	1	0	1	
12	1	1	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	1
13	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
14	1	1	1	0	1	1	1	1	1	0	1	1	0	1	1	1	1	0	1	0	1	0	1	1	1	1	0	1
15	1	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	1	0	1	

Keterangan : U adalah Pengujian, y_{x0} adalah Sa0 pada jalur x dan y_{x1} adalah Sa1 pada jalur x

Berdasarkan Tabel 3.2 didapatkan 10 kelompok kesalahan tunggal f_x dengan tujuh kelompok kesalahan memiliki pola keluaran yang sama yaitu $\{y_{10}, y_{51}, y_{61}\}$ dikelompokkan sebagai f_1 , $\{y_{11}, y_{50}, y_{60}\}$ dikelompokkan sebagai f_2 , $\{y_{20}, y_{100}\}$ dikelompokkan sebagai f_3 , $\{y_{30}, y_{90}, y_{110}\}$ menjadi kelompok f_5 , $\{y_{40}, y_{71}, y_{81}\}$ sebagai kelompok f_7 , $\{y_{41}, y_{70}, y_{80}\}$ menjadi kelompok f_8 , dan $\{y_{101}, y_{111}\}$ menjadi kelompok f_{10} serta tiga kelompok kesalahan yang memiliki pola keluaran yang berbeda yaitu $\{y_{21}\}$ sebagai kelompok f_4 , $\{y_{31}\}$ sebagai f_6 , dan $\{y_{91}\}$ menjadi f_9 seperti yang ditunjukkan pada Tabel 3.3

Tabel 3.3 Kelompok Kesalahan Potongan Rangkaian Uji Pertama

Uji	f_0 y_0	f_1 y_{10} y_{51} y_{61}	f_2 y_{11} y_{50} y_{60}	f_3 y_{20} y_{100}	f_4 y_{21}	f_5 y_{30} y_{90} y_{110}	f_6 y_{31}	f_7 y_{40} y_{71} y_{81}	f_8 y_{41} y_{70} y_{80}	f_9 y_{91}	f_{10} y_{101} y_{111}
0	0	0	0	0	1	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	1
2	0	0	1	0	1	0	0	0	0	1	1
3	0	0	0	0	0	0	0	0	0	0	1
4	1	1	0	0	1	1	1	1	0	1	1
5	0	0	0	0	0	0	0	1	0	0	1
6	1	1	1	0	1	1	1	1	0	1	1
7	0	0	0	0	0	0	0	1	0	0	1
8	0	0	0	0	0	0	1	0	0	0	1
9	0	0	0	0	0	0	0	0	0	0	1
10	1	0	1	1	1	0	1	1	0	1	1
11	0	0	0	0	0	0	0	1	0	0	1
12	0	1	0	0	0	0	1	0	0	0	1
13	0	0	0	0	0	0	0	0	0	0	1
14	1	1	1	1	1	0	1	1	0	1	1
15	0	0	0	0	0	0	0	1	0	0	1

Rangkaian uji kedua menghasilkan keluaran dengan kondisi normal atau tanpa adanya kesalahan yaitu y_0 serta kondisi jalur *input* A, B2, C2 dan D mengalami gangguan Sa0 dan Sa1 berturut-turut. Hasil tersebut didapatkan melalui Persamaan 3.2 bisa dilihat pada Tabel 3.4.

Tabel 3.4 Tabel Kesalahan Diagnosa Semua Kesalahan Tunggal dari Potongan 2 Rangkaian IC 74157.

U	Masukan				Nilai Keluaran																								
	D	C	B2	A2	y ₀	y ₁₀	y ₁₁	y ₁₂₀	y ₁₂₁	y ₁₃₀	y ₁₃₁	y ₄₀	y ₄₁	y ₅₀	y ₅₁	y ₁₄₀	y ₁₄₁	y ₁₅₀	y ₁₅₁	y ₁₆₀	y ₁₆₁	y ₁₇₀	y ₁₇₁	y ₁₈₀	y ₁₈₁	y ₁₉₀	y ₁₉₁		
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	
1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	
2	0	0	1	0	0	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	0	1	0	1	
3	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	
4	0	1	0	0	1	1	0	0	1	1	1	1	0	0	1	0	1	0	1	0	1	0	1	1	1	0	1	1	1
5	0	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	1	0	1	1	
6	0	1	1	0	1	1	1	0	1	1	1	1	0	1	1	1	1	0	1	0	1	1	1	1	0	1	1	1	
7	0	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	1	0	1	1	
8	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	
9	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	
10	1	0	1	0	1	0	1	1	1	0	1	1	0	1	0	1	0	0	1	0	1	0	1	1	1	1	0	1	
11	1	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	1	0	1	1	
12	1	1	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	1	
13	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	
14	1	1	1	0	1	1	1	1	1	0	1	1	0	1	1	1	1	0	1	0	1	0	1	1	1	1	0	1	
15	1	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	1	0	1	1	

Keterangan : U adalah Pengujian, y_{x0} adalah Sa0 pada jalur x dan y_{x1} adalah Sa1 pada jalur x

Berdasarkan Tabel 3.4 didapatkan 10 kelompok kesalahan tunggal yaitu $\{y_{10}, y_{51}, y_{141}\}$ dikelompokkan sebagai f_{11} , $\{y_{11}, y_{50}, y_{140}\}$ dikelompokkan sebagai f_{12} , $\{y_{120}, y_{180}\}$ dikelompokkan sebagai f_{13} , $\{y_{121}\}$ sebagai kelompok f_{14} , $\{y_{130}, y_{140}, y_{180}\}$ menjadi kelompok f_{15} , $\{y_{131}\}$ sebagai f_{16} , $\{y_{40}, y_{151}, y_{161}\}$ sebagai kelompok f_{17} , $\{y_{41}, y_{150}, y_{160}\}$ menjadi kelompok f_{18} , $\{y_{171}\}$ menjadi f_{19} dan $\{y_{181}, y_{191}\}$ menjadi kelompok f_{20} .

Tabel 3.5 Pengelompokkan Kesalahan Pada Rangkaian Uji 2

Uji	f_0 y_0	f_{11} y_{10} y_{51} y_{141}	F_{12} y_{11} y_{50} y_{140}	f_{13} y_{120} y_{180}	f_{14} y_{121}	f_{15} y_{130} y_{170} y_{190}	f_{16} y_{131}	f_{17} y_{40} y_{151} y_{161}	f_{18} y_{41} y_{150} y_{160}	f_{19} y_{171}	F_{20} y_{181} y_{191}
0	0	0	0	0	1	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	1
2	0	0	1	0	1	0	0	0	0	1	1
3	0	0	0	0	0	0	0	0	0	0	1
4	1	1	0	0	1	1	1	1	0	1	1
5	0	0	0	0	0	0	0	1	0	0	1
6	1	1	1	0	1	1	1	1	0	1	1
7	0	0	0	0	0	0	0	1	0	0	1
8	0	0	0	0	0	0	1	0	0	0	1
9	0	0	0	0	0	0	0	0	0	0	1
10	1	0	1	1	1	0	1	1	0	1	1
11	0	0	0	0	0	0	0	1	0	0	1
12	0	1	0	0	0	0	1	0	0	0	1
13	0	0	0	0	0	0	0	0	0	0	1
14	1	1	1	1	1	0	1	1	0	1	1
15	0	0	0	0	0	0	0	1	0	0	1

Rangkaian uji 3 menghasilkan keluaran dengan kondisi normal atau tanpa adanya kesalahan yaitu y_0 serta kondisi jalur *input* A, B3, C3 dan D mengalami gangguan Sa0 dan Sa1 berturut-turut. Hasil tersebut didapatkan melalui Persamaan 3.3 bisa dilihat pada Tabel 3.6.

Tabel 3.6 Tabel Kesalahan Diagnosa Semua Kesalahan Tunggal dari Potongan 3 Rangkaian IC 74157.

U	Masukan				Nilai Keluaran																								
	D	C	B3	A3	y ₀	y ₁₀	y ₁₁	y ₂₀₀	y ₂₀₁	y ₂₁₀	y ₂₁₁	y ₄₀	y ₄₁	y ₅₀	y ₅₁	y ₂₂₀	y ₂₂₁	y ₂₃₀	y ₂₃₁	y ₂₄₀	y ₂₄₁	y ₂₅₀	y ₂₅₁	y ₂₆₀	y ₂₆₁	y ₂₇₀	y ₂₇₁		
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	
1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	
2	0	0	1	0	0	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	0	1	0	1	
3	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	
4	0	1	0	0	1	1	0	0	1	1	1	1	0	0	1	0	1	0	1	0	1	0	1	1	1	0	1	1	1
5	0	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	1	0	1	1	
6	0	1	1	0	1	1	1	0	1	1	1	1	0	1	1	1	1	0	1	0	1	1	1	1	0	1	1	1	
7	0	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	1	0	1	1	
8	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	
9	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	
10	1	0	1	0	1	0	1	1	1	0	1	1	0	1	0	1	0	0	1	0	1	0	1	1	1	1	0	1	
11	1	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	1	0	1	1	
12	1	1	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	1	
13	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	
14	1	1	1	0	1	1	1	1	1	0	1	1	0	1	1	1	1	0	1	0	1	0	1	1	1	1	0	1	
15	1	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	1	0	1	1	

Keterangan : U adalah Pengujian, y_{x0} adalah Sa0 pada jalur x dan y_{x1} adalah Sa1 pada jalur x

Berdasarkan Tabel 3.6 didapatkan 10 kelompok kesalahan yaitu $\{y_{10}, y_{51}, y_{221}\}$ dikelompokkan sebagai f_{21} , $\{y_{11}, y_{50}, y_{220}\}$ dikelompokkan sebagai f_{22} , $\{y_{200}, y_{260}\}$ dikelompokkan sebagai f_{23} , $\{y_{201}\}$ sebagai kelompok f_{24} , $\{y_{210}, y_{250}, y_{270}\}$ menjadi kelompok f_{25} , $\{y_{211}\}$ sebagai f_{26} , $\{y_{40}, y_{231}, y_{241}\}$ sebagai kelompok f_{27} , $\{y_{41}, y_{230}, y_{240}\}$ menjadi kelompok f_{28} , $\{y_{251}\}$ menjadi f_{29} dan $\{y_{261}, y_{271}\}$ menjadi kelompok f_{30} .

Tabel 3.7 Pengelompokkan Kesalahan Pada Rangkaian Uji 3.

Uji	f_0 y_0	f_{21} y_{10} y_{51} y_{221}	f_{22} y_{11} y_{50} y_{220}	f_{23} y_{200} y_{260}	f_{24} y_{201}	f_{25} y_{210} y_{250} y_{270}	f_{26} y_{211}	f_{27} y_{40} y_{231} y_{241}	f_{28} y_{41} y_{230} y_{240}	f_{29} y_{251}	f_{30} y_{261} y_{271}
0	0	0	0	0	1	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	1
2	0	0	1	0	1	0	0	0	0	1	1
3	0	0	0	0	0	0	0	0	0	0	1
4	1	1	0	0	1	1	1	1	0	1	1
5	0	0	0	0	0	0	0	1	0	0	1
6	1	1	1	0	1	1	1	1	0	1	1
7	0	0	0	0	0	0	0	1	0	0	1
8	0	0	0	0	0	0	1	0	0	0	1
9	0	0	0	0	0	0	0	0	0	0	1
10	1	0	1	1	1	0	1	1	0	1	1
11	0	0	0	0	0	0	0	1	0	0	1
12	0	1	0	0	0	0	1	0	0	0	1
13	0	0	0	0	0	0	0	0	0	0	1
14	1	1	1	1	1	0	1	1	0	1	1
15	0	0	0	0	0	0	0	1	0	0	1

Rangkaian uji 4 menghasilkan keluaran dengan kondisi normal atau tanpa adanya kesalahan yaitu y_0 serta kondisi jalur *input* A, B4, C4 dan D mengalami gangguan Sa0 dan Sa1 berturut-turut. Hasil tersebut didapatkan melalui Persamaan 3.4 bisa dilihat pada Tabel 3.8..

Tabel 3.8 Tabel Kesalahan Diagnosa Semua Kesalahan Tunggal dari Potongan 4 Rangkaian IC 74157.

U	Masukan				Nilai Keluaran																							
	D	C	B4	A4	y ₀	y ₁₀	y ₁₁	y ₂₈₀	y ₂₈₁	y ₂₉₀	y ₂₉₁	y ₄₀	y ₄₁	y ₅₀	y ₅₁	y ₃₀₀	y ₃₀₁	y ₃₁₀	y ₃₁₁	y ₃₂₀	y ₃₂₁	y ₃₃₀	y ₃₃₁	y ₃₄₀	y ₃₄₁	y ₃₅₀	y ₃₅₁	
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
2	0	0	1	0	0	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	0	1	0	1
3	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
4	0	1	0	0	1	1	0	0	1	1	1	1	0	0	1	0	1	0	1	0	1	1	1	0	1	1	1	1
5	0	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	1	0	1	
6	0	1	1	0	1	1	1	0	1	1	1	1	0	1	1	1	1	0	1	0	1	1	1	0	1	1	1	
7	0	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	1	0	1	
8	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
9	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
10	1	0	1	0	1	0	1	1	1	0	1	1	0	1	0	1	0	0	1	0	1	0	1	1	1	1	0	1
11	1	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	1	0	1	
12	1	1	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0	1	0	1	
13	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	
14	1	1	1	0	1	1	1	1	1	0	1	1	0	1	1	1	1	0	1	0	1	0	1	1	1	1	0	1
15	1	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	1	0	1	

Keterangan : U adalah Pengujian, y_{x0} adalah Sa0 pada jalur x dan y_{x1} adalah Sa1 pada jalur x

Berdasarkan Tabel 3.8 didapatkan kelompok kesalahan yaitu f_0 tanpa adanya kesalahan, $\{y_{10}, y_{51}, y_{301}\}$ dikelompokkan sebagai f_{31} , $\{y_{11}, y_{50}, y_{300}\}$ dikelompokkan sebagai f_{32} , $\{y_{280}, y_{340}\}$ dikelompokkan sebagai f_{33} , $\{y_{281}\}$ sebagai kelompok f_{34} , $\{y_{290}, y_{330}, y_{350}\}$ menjadi kelompok f_{35} , $\{y_{291}\}$ sebagai f_{36} , $\{y_{40}, y_{311}, y_{321}\}$ sebagai kelompok f_{37} , $\{y_{41}, y_{310}, y_{320}\}$ menjadi kelompok f_{38} , $\{y_{331}\}$ menjadi f_{39} dan $\{y_{341}, y_{351}\}$ menjadi kelompok f_{40} .

Tabel 3.9 Pengelompokkan Kesalahan Pada Rangkaian Uji 4.

Uji	f_0 y_0	f_{31} y_{10} y_{51} y_{301}	f_{32} y_{11} y_{50} y_{300}	f_{33} y_{280} y_{340}	f_{34} y_{281}	f_{35} y_{290} y_{330} y_{350}	f_{36} y_{291}	f_{37} y_{40} y_{311} y_{321}	f_{38} y_{41} y_{310} y_{320}	f_{39} y_{331}	f_{40} y_{341} y_{351}
0	0	0	0	0	1	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	1
2	0	0	1	0	1	0	0	0	0	1	1
3	0	0	0	0	0	0	0	0	0	0	1
4	1	1	0	0	1	1	1	1	0	1	1
5	0	0	0	0	0	0	0	1	0	0	1
6	1	1	1	0	1	1	1	1	0	1	1
7	0	0	0	0	0	0	0	1	0	0	1
8	0	0	0	0	0	0	1	0	0	0	1
9	0	0	0	0	0	0	0	0	0	0	1
10	1	0	1	1	1	0	1	1	0	1	1
11	0	0	0	0	0	0	0	1	0	0	1
12	0	1	0	0	0	0	1	0	0	0	1
13	0	0	0	0	0	0	0	0	0	0	1
14	1	1	1	1	1	0	1	1	0	1	1
15	0	0	0	0	0	0	0	1	0	0	1

3.6.2 Penentuan Set Pengujian Lengkap Minimum

Set pengujian lengkap minimum merupakan set pengujian lengkap yang berisikan pengujian dengan jumlah minimum. Set pengujian lengkap untuk deteksi setiap kesalahan didapatkan dengan cara membandingkan antar kolom kelompok kesalahan menggunakan operasi X-OR. Hasil dari operasi X-OR antar kelompok kesalahan seperti yang di tunjukan pada Tabel 3.10. Selanjutnya untuk

mendapatkan set minimum dilakukan dengan cara menghilangkan kolom dan baris yang berlebihan pada Tabel 3.10 dengan cara:

- Menghilangkan setiap baris yang memiliki 1 dalam kolom yang identik dengan 1 dalam beberapa baris yang lain, yaitu dengan menghilangkan setiap baris yang dicakup oleh beberapa baris lain seperti pada Gambar 3.5.

Uji	X-OR Set Kesalahan														
	f	0	0	0	0	0	f	1	1	1	1	f	2	2	2
		1	2	3	4	5		2	3	4	5		3	4	5
0					1					1				1	
1															
2			1		1			1		1			1		1
3															
4			1		1			1		1			1		1
5															

Gambar 3.5 Menghilangkan Baris

- Menghilangkan setiap kolom yang memiliki 1 dalam semua baris tempat kolom lain yang memiliki 1, yaitu menghilangkan setiap kolom yang mencakup beberapa kolom lain. Seperti pada Tabel 3.5. Langkah tersebut dilakukan berulang hingga tidak dapat lagi diterapkan.

Uji	X-OR Set Kesalahan														
	f	0	0	0	0	0	f	1	1	1	1	f	2	2	2
		1	2	3	4	5		2	3	4	5		3	4	5
0					1					1				1	
1															
2			1		1			1		1			1		1
3															
4			1		1			1		1			1		1
5															

Gambar 3.6 Menghilangkan Baris dan Kolom

Tabel 3.10 Penentuan Set Pengujian Lengkap Minimum

Uji	Hasil X-OR Set Kesalahan														
	f	0 0 0 0 0 0 0 0 0 0	1 1 1 1 1 1 1 1 1	2 2 2 2 2 2 2 2	3 3 3 3 3 3 3	4 4 4 4 4 4	5 5 5 5 5	6 6 6 6	7 7 7	8 8	9				
	1 2 3 4 5 6 7 8 9 10	2 3 4 5 6 7 8 9 10	3 4 5 6 7 8 9 10	4 5 6 7 8 9 10	5 6 7 8 9 10	6 7 8 9 10	7 8 9 10	8 9 10	9 10	10					
0		1		1	1		1	1 1 1 1	1		1	1	1	1	
1											1		1	1	
2		1	1			1	1	1 1 1 1	1		1	1	1	1	
3											1		1	1	
4		1	1		1	1 1		1 1 1 1	1 1		1		1	1	
5					1		1	1	1	1	1	1	1	1	
6		1			1	1	1	1	1	1	1	1	1	1	
7				1		1	1	1	1	1	1	1	1	1	
8				1		1	1	1	1	1	1	1	1	1	
9						1	1	1	1	1	1	1	1	1	
10	1		1		1	1 1 1	1 1	1 1	1	1	1	1 1	1 1	1	
11				1		1	1	1	1	1	1	1	1	1	
12	1			1		1 1 1 1	1 1 1		1	1	1	1	1 1 1	1	
13						1		1	1	1	1	1	1	1	
14			1		1		1	1	1	1	1	1 1	1 1	1	
15				1		1	1	1	1	1	1	1	1 1	1	

Keterangan: $f_{ab} = f_a \text{ X-OR } f_b$

Set pengujian lengkap minimum pada penelitian ini tersusun dari tiga pengujian yaitu set pengujian penting primer, set pengujian penting sekunder, dan set pengujian tidak penting. Untuk menentukan pengujian penting primer yaitu dengan memilih baris pada kolom yang memiliki hanya satu hasil operasi X-OR dari perbandingan antara kelompok kesalahan F_0 dengan kelompok kesalahan $f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9,$ dan f_{10} pada potongan 1 rangkaian IC 74157. Berdasarkan Tabel 3.10 hasil operasi X-OR hanya ada satu pada perbandingan f_0 dan f_9 saat pengujian (2). Sehingga didapatkan pengujian penting primer yaitu pengujian (2), selanjutnya melakukan langkah seperti pada Gambar 3.5 dan Gambar 3.6. Hasilnya pengujian penting primer (2) mencakup hampir seluruh baris dan kolom kesalahan. Proses tersebut bisa dilihat pada Gambar 3.7.

Proses selanjutnya yaitu penentuan pengujian penting sekunder. Pengujian penting sekunder didapatkan dengan memilih baris pada kolom yang memiliki hasil paling sedikit dari operasi X-OR dari perbandingan antara kelompok kesalahan F_0 dengan kelompok kesalahan $f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9,$ dan f_{10} . Hasil paling sedikit dari operasi X-OR yaitu dua pada pengujian (10). Langkah selanjutnya yaitu seperti pada Gambar 3.4 dan 3.5 sehingga didapatkan pengujian penting sekunder pertama yaitu pada pengujian (10), proses tersebut bisa dilihat pada Gambar 3.8.

Uji	Hasil X-OR antara F																											
	f	0	0	0	0	0	0	1	1	1	1	1	2	2	2	3	3	3	3	4	4	5	5	5	6	6	7	9
	1	3	5	6	7	8	3	5	6	7	8	4	9	10	5	6	7	8	9	10	6	7	8	7	8	8	10	
0												1	1						1									1
1														1						1								1
3														1						1								1
4		1					1	1				1	1	1	1	1	1					1		1	1	1		
5					1						1			1			1			1		1		1	1	1		1
6		1					1	1			1					1	1	1				1		1	1	1		1
7					1						1			1			1			1		1		1	1	1		1
8						1					1			1		1				1		1		1	1	1		1
9														1						1		1		1	1	1		1
10	1													1						1		1		1	1	1		1
11						1							1			1				1		1		1	1	1		1
12	1				1			1	1		1		1		1	1				1	1	1		1	1	1		1
13													1							1								1
14									1		1				1					1		1	1		1	1		1
15						1							1			1				1		1		1	1	1		1

Keterangan: $f_{ab} = f_a \text{ X-OR } f_b$

Gambar 3.8 Penentuan Pengujian Penting Sekunder (10)

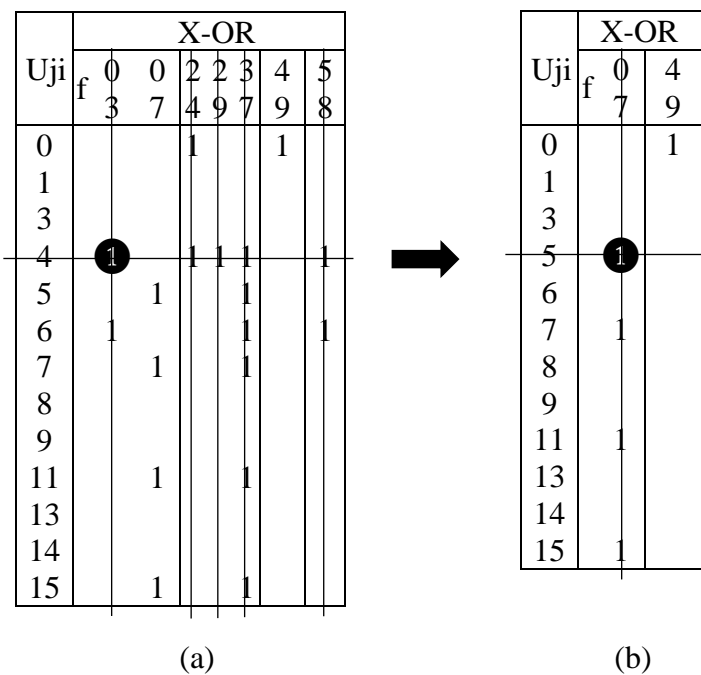
Proses selanjutnya yaitu penentuan pengujian penting sekunder ke-2. Pengujian (12) dipilih karena memiliki hasil paling sedikit dari operasi X-OR antara kelompok kesalahan f_0 dengan kelompok kesalahan lainnya. Proses ini dapat dilihat pada Gambar 3.9.

Uji	Hasil X-OR antara F														
	f	0	0	0	1	1	2	2	2	3	3	4	4	5	6
	3	6	7	5	8	4	9	10	6	7	9	10	8	7	10
0						1					1				1
1								1					1		1
3									1				1		1
4	1				1	1	1		1	1				1	
5				1						1			1		1
6	1				1					1	1			1	
7				1							1			1	1
8		1							1				1	1	1
9													1	1	1
11				1						1				1	1
12		1			1	1			1	1				1	1
13														1	1
14					1	1								1	1
15				1					1		1			1	1

Keterangan: $f_{ab} = f_a \text{ X-OR } f_b$

Gambar 3.9 Penentuan Pengujian Penting Sekunder (12)

Pengujian (4) dipilih sebagai pengujian penting sekunder ke-3 dengan langkah yang sama seperti sebelumnya. Proses tersebut bisa dilihat pada Gambar 3.10 (a).



Gambar 3.10 (a) Penentuan Pengujian (4), (b) Penentuan Pengujian (5)

Tersisa satu kolom hasil operasi X-OR pengujian penting sekunder yaitu pada kelompok kesalahan F_0 dengan kelompok kesalahan F_7 . Sehingga pengujian penting sekunder terakhir adalah pengujian (5) seperti pada Gambar 3.10 (b).

Uji	X-OR	
	f	4 9
0		1
1		
3		
6		
7		
8		
9		
11		
13		
14		
15		

Gambar 3.11 Penentuan Pengujian Tidak Penting (0)

Proses selanjutnya yaitu penentuan pengujian tidak penting. Setelah set pengujian penting didapatkan, hanya tersisa satu hasil operasi X-OR yaitu pada kelompok kesalahan f_4 dengan kelompok kesalahan f_9 di pengujian (0). Sehingga didapatkan pengujian tidak penting pada penelitian yaitu pengujian (0), proses tersebut bisa dilihat pada Gambar 3.11.

Setelah pengujian penting primer, sekunder, dan pengujian tidak penting pada potongan rangkaian uji pertama didapatkan. Maka terbentuklah set pengujian lengkap minimum yang sama pada potongan rangkaian ke-2, ke-3, dan ke4 pada penelitian ini yaitu pengujian {2,10,12,4,5,0}. Hanya kelompok kesalahan tiap potongan rangkaian uji saja yang berbeda. Cara yang efektif untuk menampilkan set pengujian lengkap minimum dengan hasilnya yaitu dengan menggunakan pohon diagnosa.

3.6.3 Pohon Diagnosa

Pohon diagnosa dihasilkan dengan penjadwalan pengujian yang bertujuan untuk mendapatkan pengujian dengan level minimal dari set pengujian lengkap minimum. Matrik F^* disusun dari set pengujian lengkap minimum, kemudian dihitung selisih jumlah pasangan 0 dan 1 menggunakan persamaan 2.2 seperti pada Gambar 3.12. Baris pengujian (2) dipilih karena memiliki selisih terkecil pasangan (0,1) yaitu 3 jika dibandingkan dengan baris lainnya.

$$F^* =$$

f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	U	Ri
0	0	1	0	1	0	0	0	0	1	1	2	3
1	0	1	1	1	0	1	1	0	1	1	10	5
0	1	0	0	0	0	1	0	0	0	1	12	5
1	1	0	0	1	1	1	1	0	1	1	4	5
0	0	0	0	0	0	0	1	0	0	1	5	7
0	0	0	0	1	0	0	0	0	0	1	0	7

U : Pengujian

Ri : Selisih terkecil jumlah (0,1) dalam baris

Gambar 3.12 Pemilihan Pengujian F^*

Pemilihan pengujian (2) menghasilkan dua urutan yaitu $F^*_{0\{2\}}$ dan $F^*_{1\{2\}}$. $F^*_{0\{2\}}$ dipilih baris pengujian (10) dengan $Ri=1$ dan matrik $F^*_{1\{2\}}$ dipilih baris pengujian (0) karena memiliki selisih jumlah 0 dan satu $Ri=0$.

$F^*(2)_0 =$								$F^*(2)_1 =$						
f_0	f_1	f_3	f_5	f_6	f_7	f_8	U	Ri	f_2	f_4	f_9	f_{10}	U	Ri
1	0	1	0	1	1	0	10	1	1	1	1	1	10	4
0	1	0	0	1	0	0	12	3	0	0	0	1	12	2
1	1	0	1	1	1	0	4	3	0	1	1	1	4	2
0	0	0	0	0	1	0	5	5	0	0	0	1	5	2
0	0	0	0	0	0	0	0	7	0	1	0	1	0	0

(a)

(b)

Gambar 3.13 (a) Pengujian $F^*(2)_0$ dan (b) Pengujian $F^*(2)_1$

Berdasarkan Gambar 3.13(a) baris pengujian yang memiliki selisih jumlah angka 0 dan 1 paling sedikit adalah pengujian (10). Dari $F^*_{0\{2\}}$

menghasilkan dua sub urutan berikutnya yaitu $F^*_{00}\{2,10\}$ bisa dilihat pada Gambar 3.14(a) dan $F^*_{01}\{2,10\}$ bisa dilihat pada Gambar 3.14(b).

$F^*(10)_{00} =$ <table style="border-collapse: collapse; margin: auto;"> <tr> <td style="border-right: 1px solid black; padding: 5px;">f_1</td> <td style="padding: 5px;">f_5</td> <td style="padding: 5px;">f_8</td> <td style="border-right: 1px solid black; padding: 5px;">U</td> <td style="padding: 5px;">Ri</td> </tr> <tr> <td style="border-right: 1px solid black; padding: 5px;">1</td> <td style="padding: 5px;">0</td> <td style="padding: 5px;">0</td> <td style="border-right: 1px solid black; padding: 5px;">12</td> <td style="padding: 5px;">1</td> </tr> <tr> <td style="border-right: 1px solid black; padding: 5px;">1</td> <td style="padding: 5px;">1</td> <td style="padding: 5px;">0</td> <td style="border-right: 1px solid black; padding: 5px;">4</td> <td style="padding: 5px;">1</td> </tr> <tr> <td style="border-right: 1px solid black; padding: 5px;">0</td> <td style="padding: 5px;">0</td> <td style="padding: 5px;">0</td> <td style="border-right: 1px solid black; padding: 5px;">5</td> <td style="padding: 5px;">3</td> </tr> <tr> <td style="border-right: 1px solid black; padding: 5px;">0</td> <td style="padding: 5px;">0</td> <td style="padding: 5px;">0</td> <td style="border-right: 1px solid black; padding: 5px;">0</td> <td style="padding: 5px;">3</td> </tr> </table> <p style="text-align: center;">(a)</p>	f_1	f_5	f_8	U	Ri	1	0	0	12	1	1	1	0	4	1	0	0	0	5	3	0	0	0	0	3	$F^*(10)_{01} =$ <table style="border-collapse: collapse; margin: auto;"> <tr> <td style="border-right: 1px solid black; padding: 5px;">f_0</td> <td style="padding: 5px;">f_3</td> <td style="padding: 5px;">f_6</td> <td style="padding: 5px;">f_7</td> <td style="border-right: 1px solid black; padding: 5px;">U</td> <td style="padding: 5px;">Ri</td> </tr> <tr> <td style="border-right: 1px solid black; padding: 5px;">0</td> <td style="padding: 5px;">0</td> <td style="padding: 5px;">1</td> <td style="padding: 5px;">0</td> <td style="border-right: 1px solid black; padding: 5px;">12</td> <td style="padding: 5px;">2</td> </tr> <tr> <td style="border-right: 1px solid black; padding: 5px;">1</td> <td style="padding: 5px;">0</td> <td style="padding: 5px;">1</td> <td style="padding: 5px;">1</td> <td style="border-right: 1px solid black; padding: 5px;">4</td> <td style="padding: 5px;">2</td> </tr> <tr> <td style="border-right: 1px solid black; padding: 5px;">0</td> <td style="padding: 5px;">0</td> <td style="padding: 5px;">0</td> <td style="padding: 5px;">1</td> <td style="border-right: 1px solid black; padding: 5px;">5</td> <td style="padding: 5px;">2</td> </tr> <tr> <td style="border-right: 1px solid black; padding: 5px;">0</td> <td style="padding: 5px;">0</td> <td style="padding: 5px;">0</td> <td style="padding: 5px;">0</td> <td style="border-right: 1px solid black; padding: 5px;">0</td> <td style="padding: 5px;">4</td> </tr> </table> <p style="text-align: center;">(b)</p>	f_0	f_3	f_6	f_7	U	Ri	0	0	1	0	12	2	1	0	1	1	4	2	0	0	0	1	5	2	0	0	0	0	0	4
f_1	f_5	f_8	U	Ri																																																				
1	0	0	12	1																																																				
1	1	0	4	1																																																				
0	0	0	5	3																																																				
0	0	0	0	3																																																				
f_0	f_3	f_6	f_7	U	Ri																																																			
0	0	1	0	12	2																																																			
1	0	1	1	4	2																																																			
0	0	0	1	5	2																																																			
0	0	0	0	0	4																																																			
$F^*(0)_{10} =$ <table style="border-collapse: collapse; margin: auto;"> <tr> <td style="border-right: 1px solid black; padding: 5px;">f_2</td> <td style="padding: 5px;">f_9</td> <td style="border-right: 1px solid black; padding: 5px;">U</td> <td style="padding: 5px;">Ri</td> </tr> <tr> <td style="border-right: 1px solid black; padding: 5px;">1</td> <td style="padding: 5px;">1</td> <td style="border-right: 1px solid black; padding: 5px;">10</td> <td style="padding: 5px;">2</td> </tr> <tr> <td style="border-right: 1px solid black; padding: 5px;">0</td> <td style="padding: 5px;">0</td> <td style="border-right: 1px solid black; padding: 5px;">12</td> <td style="padding: 5px;">2</td> </tr> <tr> <td style="border-right: 1px solid black; padding: 5px;">0</td> <td style="padding: 5px;">1</td> <td style="border-right: 1px solid black; padding: 5px;">4</td> <td style="padding: 5px;">0</td> </tr> <tr> <td style="border-right: 1px solid black; padding: 5px;">0</td> <td style="padding: 5px;">0</td> <td style="border-right: 1px solid black; padding: 5px;">5</td> <td style="padding: 5px;">2</td> </tr> </table> <p style="text-align: center;">(c)</p>	f_2	f_9	U	Ri	1	1	10	2	0	0	12	2	0	1	4	0	0	0	5	2	$F^*(0)_{11} =$ <table style="border-collapse: collapse; margin: auto;"> <tr> <td style="border-right: 1px solid black; padding: 5px;">f_4</td> <td style="padding: 5px;">f_{10}</td> <td style="border-right: 1px solid black; padding: 5px;">U</td> <td style="padding: 5px;">Ri</td> </tr> <tr> <td style="border-right: 1px solid black; padding: 5px;">1</td> <td style="padding: 5px;">1</td> <td style="border-right: 1px solid black; padding: 5px;">10</td> <td style="padding: 5px;">2</td> </tr> <tr> <td style="border-right: 1px solid black; padding: 5px;">0</td> <td style="padding: 5px;">1</td> <td style="border-right: 1px solid black; padding: 5px;">12</td> <td style="padding: 5px;">1</td> </tr> <tr> <td style="border-right: 1px solid black; padding: 5px;">1</td> <td style="padding: 5px;">1</td> <td style="border-right: 1px solid black; padding: 5px;">4</td> <td style="padding: 5px;">2</td> </tr> <tr> <td style="border-right: 1px solid black; padding: 5px;">0</td> <td style="padding: 5px;">1</td> <td style="border-right: 1px solid black; padding: 5px;">5</td> <td style="padding: 5px;">1</td> </tr> </table> <p style="text-align: center;">(d)</p>	f_4	f_{10}	U	Ri	1	1	10	2	0	1	12	1	1	1	4	2	0	1	5	1															
f_2	f_9	U	Ri																																																					
1	1	10	2																																																					
0	0	12	2																																																					
0	1	4	0																																																					
0	0	5	2																																																					
f_4	f_{10}	U	Ri																																																					
1	1	10	2																																																					
0	1	12	1																																																					
1	1	4	2																																																					
0	1	5	1																																																					

Gambar 3.13 (a) Urutan $F^*_{00}\{2,10\}$, (b) Urutan $F^*_{01}\{2,10\}$, (c) Urutan $F^*_{10}\{2,0\}$, (d) Urutan $F^*_{11}\{2,0\}$ dengan 2 Level Pengujian

Urutan pengujian pohon diagnosa $F^*\{2\}_1$ memiliki selisih minimum jumlah angka 0 dan 1 yaitu nol. Selisih minimum tersebut berada pada pengujian (0) sehingga didapatkan dua sub urutan baru yaitu $F^*_{10}\{2,0\}$ bisa dilihat pada Gambar 3.13(c) dan $F^*_{11}\{2,0\}$ bisa dilihat pada Gambar 3.13(d). ke-4 sub urutan baru tersebut memiliki 2 level panjang pengujian.

Berdasarkan Gambar 3.13 didapatkan delapan sub urutan pohon diagnosa terbaru dengan 3 level panjang pengujian. Sub urutan $F^*_{00}\{2,10\}$ memiliki Ri=1 pada pengujian (12) sehingga terbentuk sub urutan $F^*_{000}\{2,10,12\}$ bisa dilihat pada Gambar 3.14(a) dan dihasilkan F_1 pada sub urutan $F^*_{001}\{2,10,12\}$ bisa dilihat pada Gambar 3.14(b). $F^*_{01}\{2,10\}$ memiliki Ri=1 pada pengujian (5) sehingga dihasilkan sub urutan $F^*_{010}\{2,10,5\}$ seperti pada Gambar 3.14(c) dan sub urutan $F^*_{011}\{2,10,5\}$ yang menghasilkan F_7 seperti pada Gambar 3.14(d).

$F^*(12)_{000} =$ <table style="border-collapse: collapse; margin: auto;"> <tr><td style="border-right: 1px solid black; padding: 5px;">f_5</td><td style="padding: 5px;">f_8</td><td style="border-right: 1px solid black; padding: 5px;">U</td><td style="padding: 5px;">Ri</td></tr> <tr><td style="border-right: 1px solid black; padding: 5px;">1</td><td style="padding: 5px;">0</td><td style="border-right: 1px solid black; padding: 5px;">4</td><td style="padding: 5px;">0</td></tr> <tr><td style="border-right: 1px solid black; padding: 5px;">0</td><td style="padding: 5px;">0</td><td style="border-right: 1px solid black; padding: 5px;">5</td><td style="padding: 5px;">2</td></tr> <tr><td style="border-right: 1px solid black; padding: 5px;">0</td><td style="padding: 5px;">0</td><td style="border-right: 1px solid black; padding: 5px;">0</td><td style="padding: 5px;">2</td></tr> </table> <p style="text-align: center;">(a)</p>	f_5	f_8	U	Ri	1	0	4	0	0	0	5	2	0	0	0	2	$F^*(12)_{001} =$ <table style="border-collapse: collapse; margin: auto;"> <tr><td style="border-right: 1px solid black; padding: 5px;">f_1</td><td style="padding: 5px;">U</td></tr> <tr><td style="border-right: 1px solid black; padding: 5px;">1</td><td style="padding: 5px;">4</td></tr> <tr><td style="border-right: 1px solid black; padding: 5px;">0</td><td style="padding: 5px;">5</td></tr> <tr><td style="border-right: 1px solid black; padding: 5px;">0</td><td style="padding: 5px;">0</td></tr> </table> <p style="text-align: center;">(b)</p>	f_1	U	1	4	0	5	0	0	$F^*(5)_{010} =$ <table style="border-collapse: collapse; margin: auto;"> <tr><td style="border-right: 1px solid black; padding: 5px;">f_0</td><td style="padding: 5px;">f_3</td><td style="border-right: 1px solid black; padding: 5px;">f_6</td><td style="padding: 5px;">U</td><td style="padding: 5px;">Ri</td></tr> <tr><td style="border-right: 1px solid black; padding: 5px;">0</td><td style="padding: 5px;">0</td><td style="border-right: 1px solid black; padding: 5px;">1</td><td style="padding: 5px;">12</td><td style="padding: 5px;">1</td></tr> <tr><td style="border-right: 1px solid black; padding: 5px;">1</td><td style="padding: 5px;">0</td><td style="border-right: 1px solid black; padding: 5px;">1</td><td style="padding: 5px;">4</td><td style="padding: 5px;">1</td></tr> <tr><td style="border-right: 1px solid black; padding: 5px;">0</td><td style="padding: 5px;">0</td><td style="border-right: 1px solid black; padding: 5px;">0</td><td style="padding: 5px;">0</td><td style="padding: 5px;">3</td></tr> </table> <p style="text-align: center;">(c)</p>	f_0	f_3	f_6	U	Ri	0	0	1	12	1	1	0	1	4	1	0	0	0	0	3
f_5	f_8	U	Ri																																											
1	0	4	0																																											
0	0	5	2																																											
0	0	0	2																																											
f_1	U																																													
1	4																																													
0	5																																													
0	0																																													
f_0	f_3	f_6	U	Ri																																										
0	0	1	12	1																																										
1	0	1	4	1																																										
0	0	0	0	3																																										
$F^*(5)_{011} =$ <table style="border-collapse: collapse; margin: auto;"> <tr><td style="border-right: 1px solid black; padding: 5px;">f_7</td><td style="padding: 5px;">U</td></tr> <tr><td style="border-right: 1px solid black; padding: 5px;">0</td><td style="padding: 5px;">12</td></tr> <tr><td style="border-right: 1px solid black; padding: 5px;">1</td><td style="padding: 5px;">4</td></tr> <tr><td style="border-right: 1px solid black; padding: 5px;">0</td><td style="padding: 5px;">0</td></tr> </table> <p style="text-align: center;">(d)</p>	f_7	U	0	12	1	4	0	0	$F^*(4)_{100} =$ <table style="border-collapse: collapse; margin: auto;"> <tr><td style="border-right: 1px solid black; padding: 5px;">f_2</td><td style="padding: 5px;">U</td></tr> <tr><td style="border-right: 1px solid black; padding: 5px;">1</td><td style="padding: 5px;">10</td></tr> <tr><td style="border-right: 1px solid black; padding: 5px;">0</td><td style="padding: 5px;">12</td></tr> <tr><td style="border-right: 1px solid black; padding: 5px;">0</td><td style="padding: 5px;">5</td></tr> </table> <p style="text-align: center;">(e)</p>	f_2	U	1	10	0	12	0	5	$F^*(4)_{101} =$ <table style="border-collapse: collapse; margin: auto;"> <tr><td style="border-right: 1px solid black; padding: 5px;">f_9</td><td style="padding: 5px;">U</td></tr> <tr><td style="border-right: 1px solid black; padding: 5px;">1</td><td style="padding: 5px;">10</td></tr> <tr><td style="border-right: 1px solid black; padding: 5px;">0</td><td style="padding: 5px;">12</td></tr> <tr><td style="border-right: 1px solid black; padding: 5px;">0</td><td style="padding: 5px;">5</td></tr> </table> <p style="text-align: center;">(f)</p>	f_9	U	1	10	0	12	0	5																				
f_7	U																																													
0	12																																													
1	4																																													
0	0																																													
f_2	U																																													
1	10																																													
0	12																																													
0	5																																													
f_9	U																																													
1	10																																													
0	12																																													
0	5																																													
$F^*(5)_{110} =$ <table style="border-collapse: collapse; margin: auto;"> <tr><td style="border-right: 1px solid black; padding: 5px;">f_4</td><td style="padding: 5px;">U</td></tr> <tr><td style="border-right: 1px solid black; padding: 5px;">1</td><td style="padding: 5px;">10</td></tr> <tr><td style="border-right: 1px solid black; padding: 5px;">0</td><td style="padding: 5px;">12</td></tr> <tr><td style="border-right: 1px solid black; padding: 5px;">1</td><td style="padding: 5px;">4</td></tr> </table> <p style="text-align: center;">(g)</p>	f_4	U	1	10	0	12	1	4	$F^*(5)_{111} =$ <table style="border-collapse: collapse; margin: auto;"> <tr><td style="border-right: 1px solid black; padding: 5px;">f_{10}</td><td style="padding: 5px;">U</td></tr> <tr><td style="border-right: 1px solid black; padding: 5px;">1</td><td style="padding: 5px;">10</td></tr> <tr><td style="border-right: 1px solid black; padding: 5px;">1</td><td style="padding: 5px;">12</td></tr> <tr><td style="border-right: 1px solid black; padding: 5px;">1</td><td style="padding: 5px;">4</td></tr> </table> <p style="text-align: center;">(h)</p>	f_{10}	U	1	10	1	12	1	4																													
f_4	U																																													
1	10																																													
0	12																																													
1	4																																													
f_{10}	U																																													
1	10																																													
1	12																																													
1	4																																													

Gambar 3.14 Urutan Pohon Diagnosa dengan 3 Level Pengujian

Sub urutan $F^*_{10}\{2,0\}$ memiliki $Ri=0$ pada pengujian (4) sehingga terbentuk sub urutan $F^*_{100}\{2,0,4\}$ yang menghasilkan f_2 bisa dilihat pada Gambar 3.14(e) dan didapatkan f_9 pada sub urutan $F^*_{101}\{2,0,4\}$ seperti Gambar 3.14(f). $F^*_{11}\{2,0\}$ memiliki $Ri=1$ pada pengujian (5) sehingga didapatkan sub urutan $F^*_{110}\{2,0,5\}$ yang menghasilkan f_4 seperti pada Gambar 3.14(g) dan dihasilkan f_{10} pada sub urutan $F^*_{111}\{2,0,5\}$ seperti pada Gambar 3.14(h). Ke-4 sub urutan terbaru tersebut memiliki kedalaman pengujian sepanjang 3 level pengujian.

$F^*(4)_{0000} =$ <table style="border-collapse: collapse; margin: auto;"> <tr><td style="border-right: 1px solid black; padding: 5px;">f_8</td><td style="padding: 5px;">U</td></tr> <tr><td style="border-right: 1px solid black; padding: 5px;">0</td><td style="padding: 5px;">5</td></tr> <tr><td style="border-right: 1px solid black; padding: 5px;">0</td><td style="padding: 5px;">0</td></tr> </table> <p style="text-align: center;">(a)</p>	f_8	U	0	5	0	0	$F^*(4)_{0001} =$ <table style="border-collapse: collapse; margin: auto;"> <tr><td style="border-right: 1px solid black; padding: 5px;">f_5</td><td style="padding: 5px;">U</td></tr> <tr><td style="border-right: 1px solid black; padding: 5px;">0</td><td style="padding: 5px;">5</td></tr> <tr><td style="border-right: 1px solid black; padding: 5px;">0</td><td style="padding: 5px;">0</td></tr> </table> <p style="text-align: center;">(b)</p>	f_5	U	0	5	0	0			
f_8	U															
0	5															
0	0															
f_5	U															
0	5															
0	0															
$F^*(12)_{0100} =$ <table style="border-collapse: collapse; margin: auto;"> <tr><td style="border-right: 1px solid black; padding: 5px;">f_0</td><td style="padding: 5px;">f_3</td><td style="border-right: 1px solid black; padding: 5px;">U</td></tr> <tr><td style="border-right: 1px solid black; padding: 5px;">0</td><td style="padding: 5px;">0</td><td style="border-right: 1px solid black; padding: 5px;">4</td></tr> <tr><td style="border-right: 1px solid black; padding: 5px;">0</td><td style="padding: 5px;">0</td><td style="border-right: 1px solid black; padding: 5px;">0</td></tr> </table> <p style="text-align: center;">(c)</p>	f_0	f_3	U	0	0	4	0	0	0	$F^*(12)_{0101} =$ <table style="border-collapse: collapse; margin: auto;"> <tr><td style="border-right: 1px solid black; padding: 5px;">f_6</td><td style="padding: 5px;">U</td></tr> <tr><td style="border-right: 1px solid black; padding: 5px;">0</td><td style="padding: 5px;">5</td></tr> <tr><td style="border-right: 1px solid black; padding: 5px;">0</td><td style="padding: 5px;">0</td></tr> </table> <p style="text-align: center;">(d)</p>	f_6	U	0	5	0	0
f_0	f_3	U														
0	0	4														
0	0	0														
f_6	U															
0	5															
0	0															

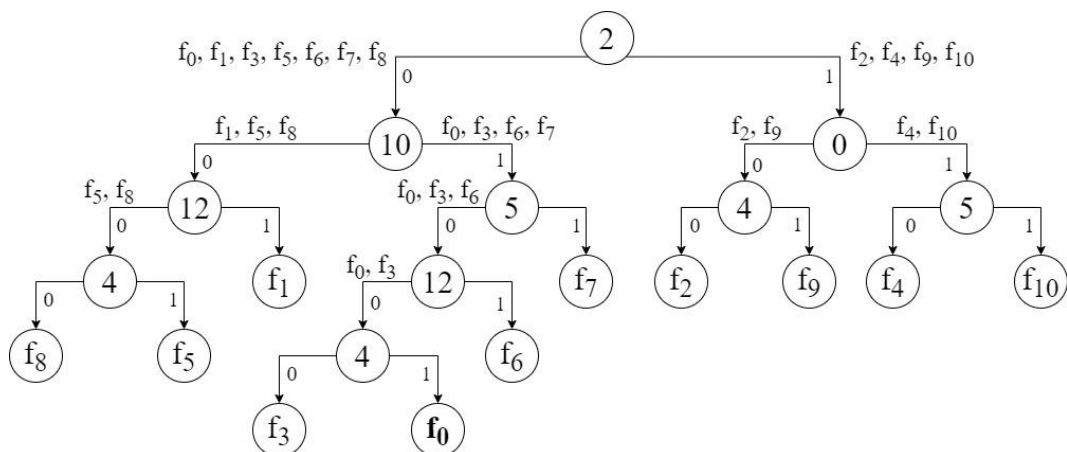
Gambar 3.15 Urutan Pohon Diagnosa dengan 4 Level Pengujian

Selanjutnya didapatkan sub urutan dengan 4 level pengujian diantaranya $F^{*0000}\{2,10,12,4\}$ yang menghasilkan f_{10} serta $F^{*0001}\{2,10,12,4\}$ menghasilkan f_5 dari penurunan urutan sebelumnya yaitu $F^{*000}\{2,10,12\}$ bisa dilihat pada Gambar 3.15 (a) dan (b). Pada urutan $F^{*010}\{2,10,5\}$ didapatkan dua sub urutan yaitu $F^{*0100}\{2,10,5,12\}$ bisa dilihat pada Gambar 3.15(c) dan $F^{*0101}\{2,10,5,12\}$ yang menghasilkan F_6 seperti pada Gambar 3.15(d).

$$\begin{array}{c}
 F^{*(4)01001} = \\
 \left| \begin{array}{c|c} f_0 & U \\ \hline 0 & 0 \end{array} \right. \\
 \text{(a)}
 \end{array}
 \qquad
 \begin{array}{c}
 F^{*(4)01000} = \\
 \left| \begin{array}{c|c} f_3 & U \\ \hline 0 & 0 \end{array} \right. \\
 \text{(b)}
 \end{array}$$

Gambar 3.16 Urutan Pohon Diagnosa dengan 5 Level Pengujian

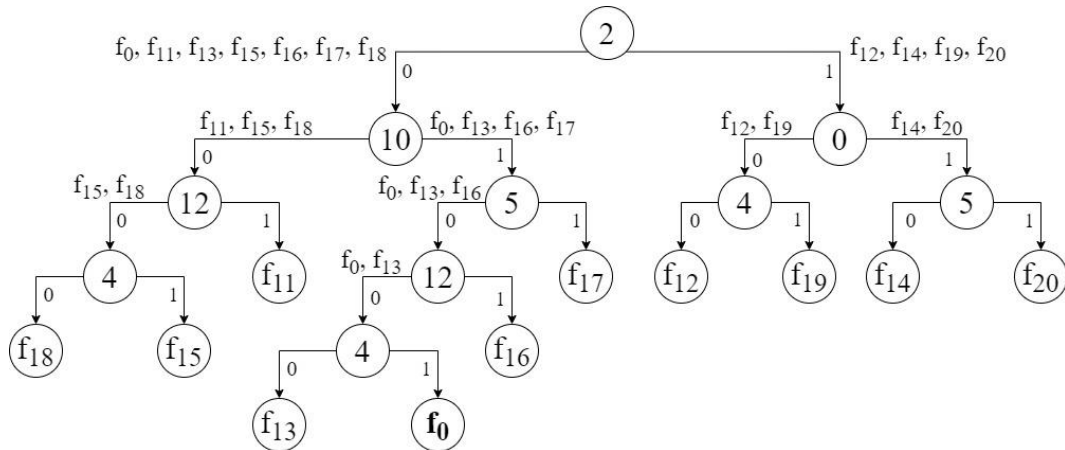
Gambar 3.16 merupakan urutan terakhir dari pohon diagnosa yang dihasilkan dari urutan sebelumnya yaitu $F^{*0100}\{2,10,5,12\}$. Sub urutan ini memiliki panjang panjang 5 level pengujian diantaranya yaitu sub urutan $F^{*01000}\{2,10,5,12,4\}$ yang menghasilkan f_3 dan sub urutan $F^{*01001}\{2,10,5,12,4\}$ yang menghasilkan f_0 . Hasil urutan tersebut disusun menjadi Pohon diagnosa seperti pada Gambar 3.17.



Gambar 3.17 Pohon Diagnosa Potongan 1 Rangkaian IC 74157.

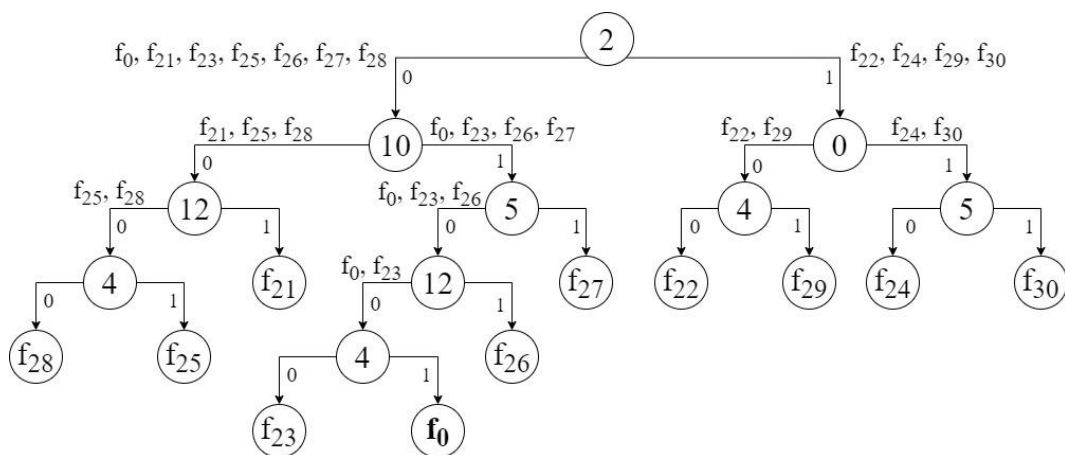
Gambar 3.18 merupakan pohon diagnosa untuk potongan 2 rangkaian uji yang dihasilkan dengan cara yang sama seperti mencari pohon diagnosa pada potongan rangkaian pertama. Pola kelompok kesalahan $\{f_1, f_2, f_3, f_4,$

$f_5, f_6, f_7, f_8, f_9,$ dan f_{10} pada rangkaian pertama identik dengan kelompok kesalahan potongan rangkaian 2 yaitu $\{f_{11}, f_{12}, f_{13}, f_{14}, f_{15}, f_{16}, f_{17}, f_{18}, f_{19},$ dan $f_{20}\}$. Sehingga lokasi kelompok kesalahan yang dihasilkan pohon diagnosa potongan rangkaian 2 sama dengan pohon diagnosa potongan 1.



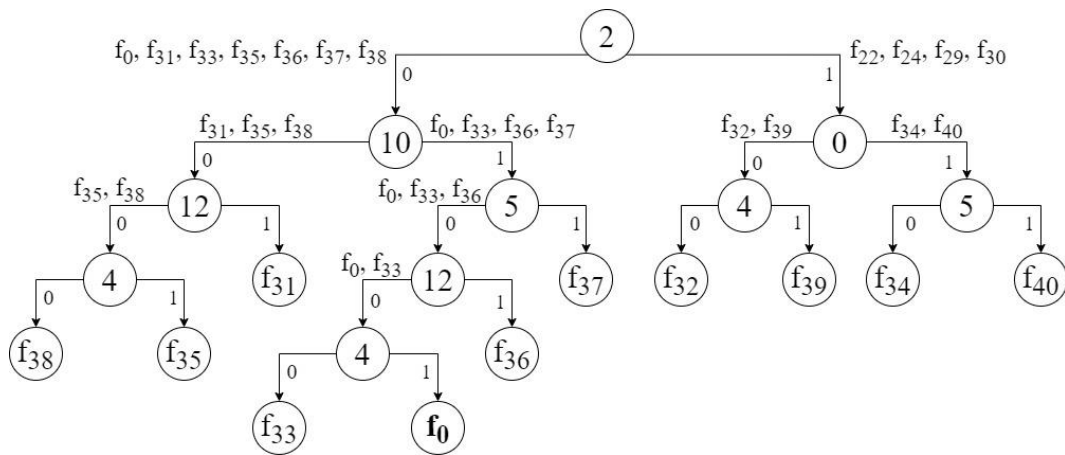
Gambar 3.18 Pohon Diagnosa Potongan 2 Rangkaian IC 74157.

Setelah pohon diagnosa rangkaian pertama dan ke-2 telah didapatkan, tahap selanjutnya yaitu membuat pohon diagnosa potongan 3 rangkaian uji dengan cara yang sama seperti pohon diagnosa sebelumnya. Lokasi kelompok kesalahan $\{f_{21}, f_{22}, f_{23}, f_{24}, f_{25}, f_{26}, f_{27}, f_{28}, f_{29},$ dan $f_{30}\}$ pada pohon diagnosa potongan rangkaian 3 identik dengan lokasi kelompok kesalahan pada pohon diagnosa sebelumnya bisa dilihat pada Gambar 3.19.



Gambar 3.19 Pohon Diagnosa Potongan 3 Rangkaian IC 74157

Setelah lokasi kelompok kesalahan pada rangkaian uji ke-3 telah berhasil dipetakan menjadi sebuah pohon diagnosa. Selanjutnya mengurutkan kelompok kesalahan $\{f_{31}, f_{32}, f_{33}, f_{34}, f_{35}, f_{36}, f_{37}, f_{38}, f_{39}, \text{ dan } f_{40}\}$ pada potongan 4 rangkaian uji. Kelompok kesalahan pada potongan 4 rangkaian uji memiliki pola yang identik dengan kelompok kesalahan pada rangkaian uji sebelumnya sehingga dihasilkan pohon diagnosa seperti pada Gambar 3.20.

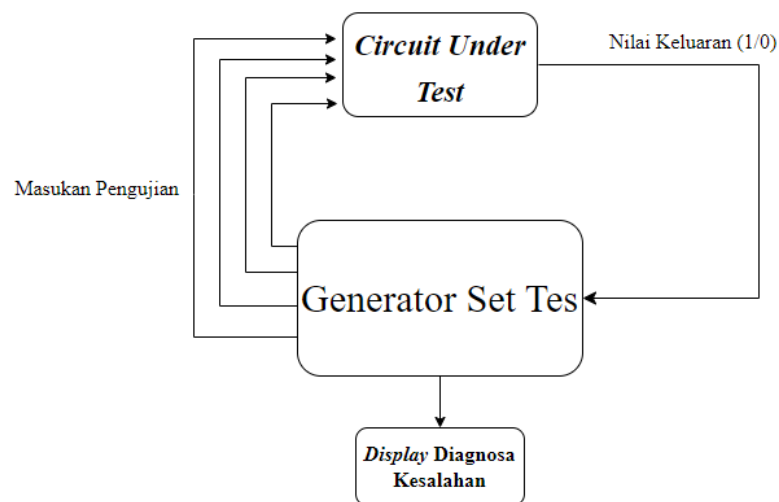


Gambar 3.20 Pohon Diagnosa Potongan 4 Rangkaian IC 74157.

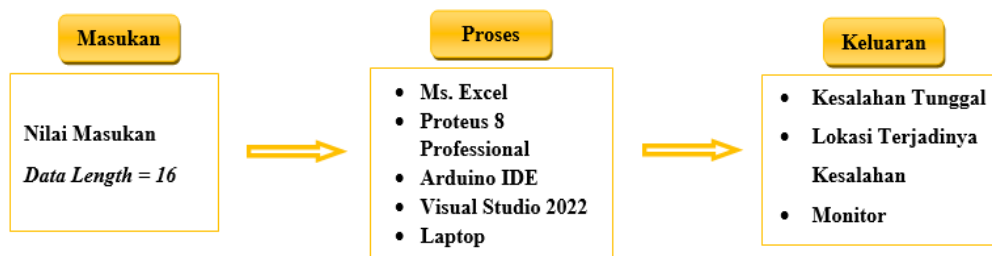
Pohon diagnosa rangkaian *multiplexer quad 2 line to 1 line* IC 74157 memiliki kedalaman pengujian terpanjang yaitu 5 level pengujian. Level pengujian adalah tingkatan cabang dari pohon diagnosa. Kedalaman pengujian merupakan gabungan tingkatan cabang pohon diagnosa yang dimulai dari level pengujian 1 sampai level pengujian 5.

3.7 Generator Set Uji

Rancangan pengujian sistem generator set uji terdiri dari empat generator rangkaian potongan IC 74157. Potongan rangkaian set uji pada penelitian ini dijabarkan dalam blok diagram pada Gambar 3.21. Pada penelitian ini data awal didapatkan dengan mengolah data pada *Microsoft Excel* menggunakan fungsi persamaan Boolean 3.1. Setelah mendapatkan data selanjutnya diprogram menggunakan perangkat lunak Arduino IDE yang akan diproses oleh Arduino nano Atmega328. Hasil dari generator set uji akan ditampilkan menggunakan GUI (*graphical user interface*) yang diciptakan melalui perangkat lunak visual studio 2022 untuk mengetahui lokasi terjadinya gangguan pada IC 74157. Susunan komponen perangkat lunak bisa dilihat pada Gambar 3.22.



Gambar 3.21 Pengaturan Eksperimen Rangkaian Set Uji

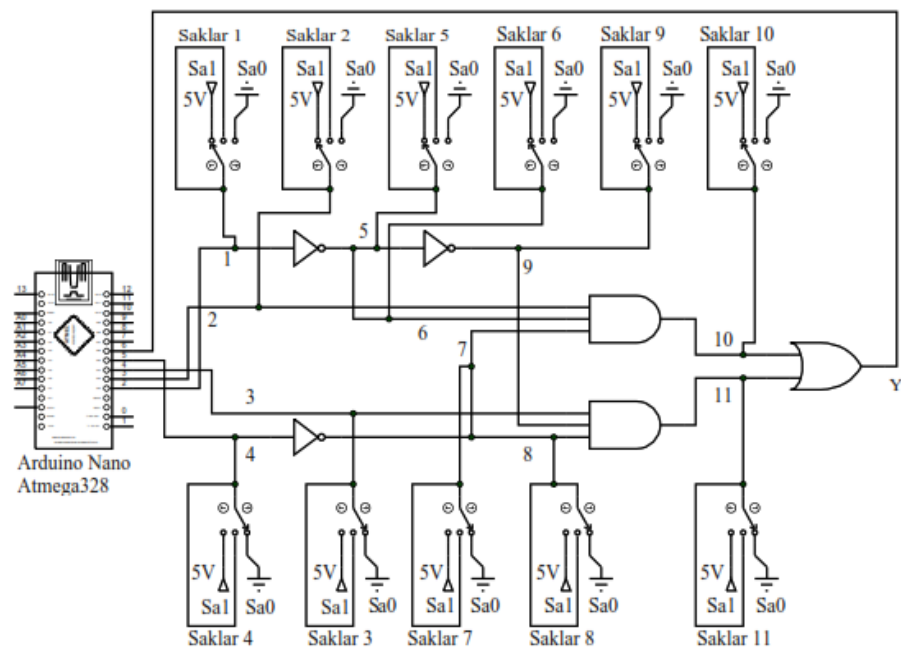


Gambar 3.22 Komponen Sistem Perangkat Lunak

3.7.1 Perancangan Pengujian

Susunan komponen dari sistem generator set uji multiplexer IC 74157 pada penelitian adalah sebagai berikut:

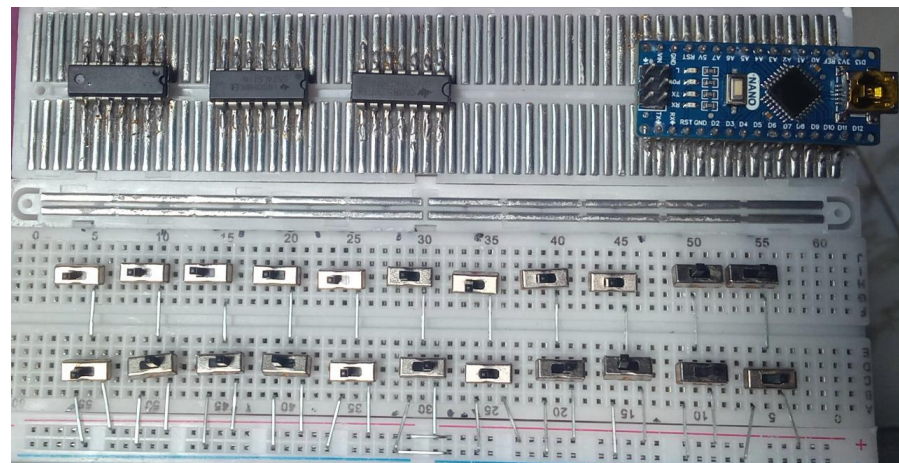
- Modul IC, Potongan rangkaian digital kombinasional IC 74157 yang dibangun dengan IC 7414 gerbang *AND*, IC 7411 gerbang *NOT*, dan IC 7432 gerbang *OR*. Penyusunan rangkaian kombinasional dengan IC pengganti ini untuk mensimulasikan gangguan *stuck* yang dianalogikan muncul pada setiap jalur masukan gerbang logika.
- Arduino nano Atmega328, generator set uji dibangun menggunakan Arduino yang berperan sebagai pusat pengontrolan untuk mengatur data dari perangkat masukan yang diteruskan ke perangkat keluaran.
- Saklar SPDT 3 posisi, digunakan sebagai pengkondisi jalur dalam kondisi normal, *stuck* 0 maupun *stuck* 1. Nomor urut saklar dipasang sesuai dengan nomor jalur masukan rangkaian kombinasional. Saklar 1 dipasang pada jalur 1, saklar 2 dipasang pada jalur 2 dan seterusnya bisa dilihat pada Gambar 3.23.



Gambar 3.23 Implementasi Potongan Rangkaian Uji Multiplexer

- Perangkat Pengolahan yaitu laptop, dalam konfigurasi, ukuran, maupun bentuk tertentu. Bertugas serta bertanggung jawab untuk menjalankan perangkat lunak yang memungkinkan pengolahan data diagnosa kesalahan yang didapatkan.

Hasil dari susunan komponen pengujian yaitu perancangan generator set uji bisa dilihat pada Gambar 3.24.



Gambar 3.24 Generator Set Uji

3.7.2 Prosedur Generator Set Uji

Pohon diagnosa IC 74157 diimplementasikan menjadi generator set uji. Arduino nano atmega328 sebagai pusat pengontrolan mengolah data hasil tersebut menggunakan perangkat lunak Arduino IDE 2.1.0. Proses diawali dengan mengolah hasil set pengujian minimum menjadi sebuah pemrograman pada Arduino IDE. Variabel masukan pada potongan IC 74157 memiliki *input* 4bit (D, C, B, A). *Input* (D) dihubungkan ke pin mode 2, *input* (C) ke pin mode 3, *input* (B) ke pin mode 4, dan *input* (A) dihubungkan ke pin mode 5. Keluaran (y) dari rangkaian potongan IC 74157 disambungkan ke pin mode 6.

Potongan IC 74157 memiliki *world length* 4bit dan memiliki data *length* sebanyak 2^4 atau 16 pengujian, dimulai dari pengujian 0 sampai pengujian

15. Dalam fungsi Arduino ide pengujian 0 sampai 15 dikodingkan sebagai berikut:

```
const int jumlahPercobaan = 16;
String nomorPercobaan;;
String percobaan[jumlahPercobaan]=
    {"0000","0001","0010","0011","0100","0101","0110","0111",
     "1000","1001","1010","1011","1100","1101","1110","1111"};
```

Setelah membuat *input* dari 16 pengujian, maka program akan membaca *word length* sesuai dengan *input* pengujian. Pengecekan Boolean diawali dengan fungsi *true* yang bertujuan untuk memeriksa apakah masih dalam proses pengecekan pohon diagnosa atau tidak, langkah tersebut dilakukan secara berulang. Hasil dari proses tersebut dikirim menjadi umpan balik ke tahap selanjutnya dengan program sebagai berikut:

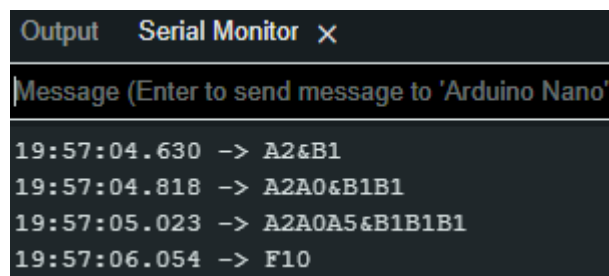
```
bool checking=true;
void pulsa(String in,int percobaan){
    for (int i=0; i<4;i++){
        if (in.charAt(i)=='0')    { logic[i]=0; }
        else{ logic[i]=1; }
    }
    digitalWrite(input1, logic[0]);
    digitalWrite(input2, logic[1]);
    digitalWrite(input3, logic[2]);
    digitalWrite(input4, logic[3]);
}
```

Program diatas sebagai penghubung *input* masing-masing pengujian ke *input logic* Arduino IDE. Jika fungsi tersebut membaca *input* 0 pada pengujian maka data yang dikirimkan ke *input logic* adalah *LOW*. Apabila *input* yang terbaca adalah 1 maka data yang dikirimkan ke *input logic* adalah *HIGH*.

Pengujian (2) menjadi awal pengujian pada generator set uji, sehingga fungsi *void pulsa* menyebutkan 0010 sebagai *input* pada pengujian (2). Berdasarkan *input* pengujian (2), maka *Input logic* 1 adalah *LOW*, *input logic* 2 adalah *LOW*, *input logic* 3 adalah *HIGH* dan *input logic* 4 adalah *LOW*. Jika, data hasil keluaran adalah 1 proses selanjutnya ke pengujian (0).

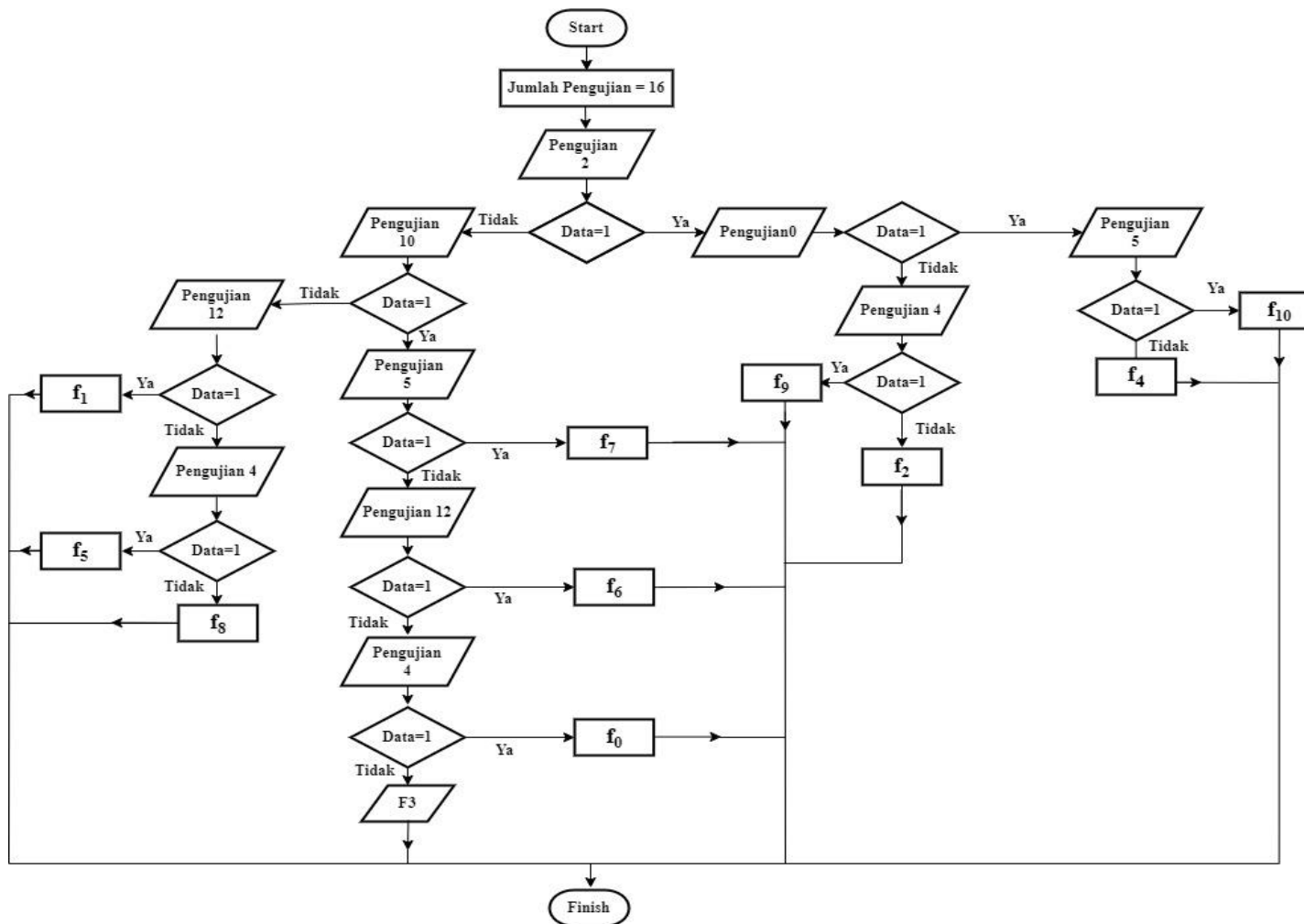
Pengujian (0) memiliki *input* yaitu 0000 sehingga data yang dikirim ke *input logic* 1,2,3 dan 4 adalah *LOW*. Jika, data hasil keluaran adalah 1 maka akan dilanjutkan ke pengujian (5). Pengujian (5) *input* yaitu 0101. Data yang dikirimkan ke *Input logic* 1 adalah *LOW*, *input logic* 2 adalah *HIGH*, *input logic* 3 adalah *LOW* dan *input logic* 4 adalah *HIGH*. Jika, data keluaran yang dihasilkan adalah 1 maka didapatkan fungsi kesalahan yaitu f_{10} pada potongan rangkaian uji pertama. Hasil dari proses pengolahan data tersebut ditampilkan secara urutan dari nomor pengujian (A) lalu data hasil keluaran (B) seperti Gambar 3.25 yang didapatkan dengan program sebagai berikut:

```
data = digitalRead(sinyalOutput);
nomorPercobaan += 'A'+String(percobaan);
output += 'B'+String(data);
outcomes = nomorPercobaan+'&'+output;
Serial.println(outcomes);
void loop() {
  if(checking){
    pulsa(percobaan[2],2);
    if(data==1){
      pulsa(percobaan[0],0);
      if(data==1){
        pulsa(percobaan[5],5);
        if(data==1){
          fault = "F10";      }
        }
      }
    }
  }
}
```



Gambar 3.25 Hasil Serial Monitor Arduino IDE Potongan Rangkaian Pertama IC 74157

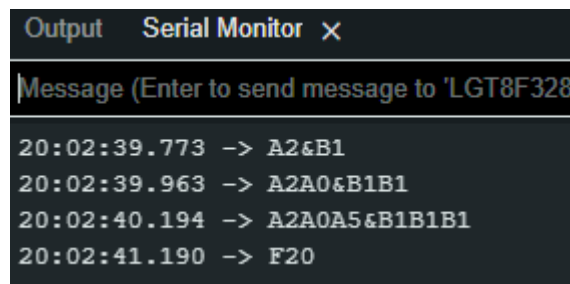
Secara keseluruhan alur dari pemrograman diagnosa kesalahan pada generator set uji potongan rangkaian pertama IC 74157 digambarkan pada diagram alir Gambar 3.26.



Gambar 3.26 Diagram Alir Coding Arduino Nano Sebagai Generator Set Uji Potongan 1 Rangkaian IC 74157

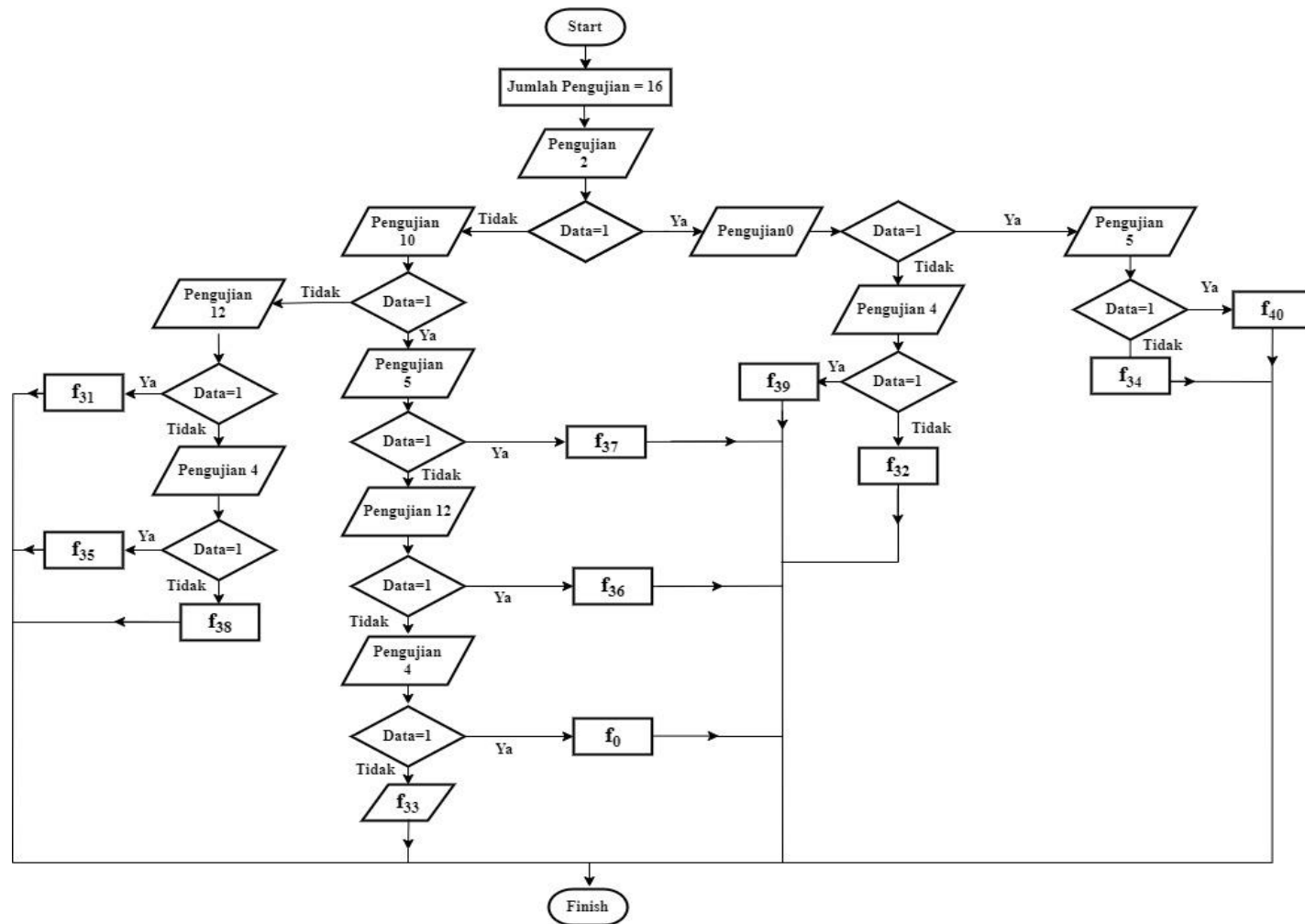
Setelah generator set uji pada potongan rangkaian pertama telah menghasilkan diagnosa kesalahan, maka generator set uji 2 akan memulai proses diagnosa pada potongan rangkaian ke-2 IC 74157. Pemrograman pada generator set uji 2 memiliki pola dan fungsi yang sama seperti generator set uji sebelumnya. Sehingga untuk mendiagnosa kelompok kesalahan yang terjadi pada potongan 2 rangkaian IC 74157 program yang digunakan sama. Hasil diagnosa kelompok kesalahan f_{20} pada potongan 2 rangkaian IC 74157 didapatkan dengan program sebagai berikut:

```
data = digitalRead(sinyalOutput);
nomorPercobaan += 'A'+String(percobaan);
output += 'B'+String(data);
outcomes = nomorPercobaan+'&'+output;
Serial.println(outcomes);
void loop() {
  if(checking){
    pulsa(percobaan[2],2);
    if(data==1){
      pulsa(percobaan[0],0);
      if(data==1){
        pulsa(percobaan[5],5);
        if(data==1){
          fault = "F20";      }
        }
      }
    }
  }
}
```



Gambar 3.27 Hasil Serial Monitor Arduino IDE Potongan 2 Rangkaian IC 74157

Gambar 3.28 merupakan diagram alir dari proses pemrograman generator set uji pada potongan 2 rangkaian IC 74157 yang memiliki pola yang sama dengan potongan sebelumnya.



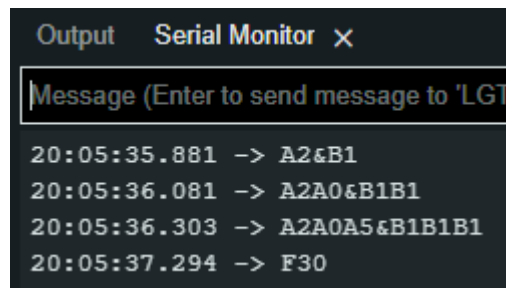
Gambar 3.28 Diagram Alir *Coding* Arduino Nano Sebagai Generator Set Uji Potongan 2 Rangkaian IC 74157

Generator set uji pada potongan rangkaian ke-2 berhasil mendiagnosa gangguan, selanjutnya generator set uji pada potongan 3 rangkaian uji IC 74157 akan memulai mendiagnosa gangguan. Generator uji 3 dibangun menggunakan program yang identik dengan generator sebelumnya. Misalnya diagnosa gangguan f₃₀ dibuat menggunakan program sebagai berikut:

```

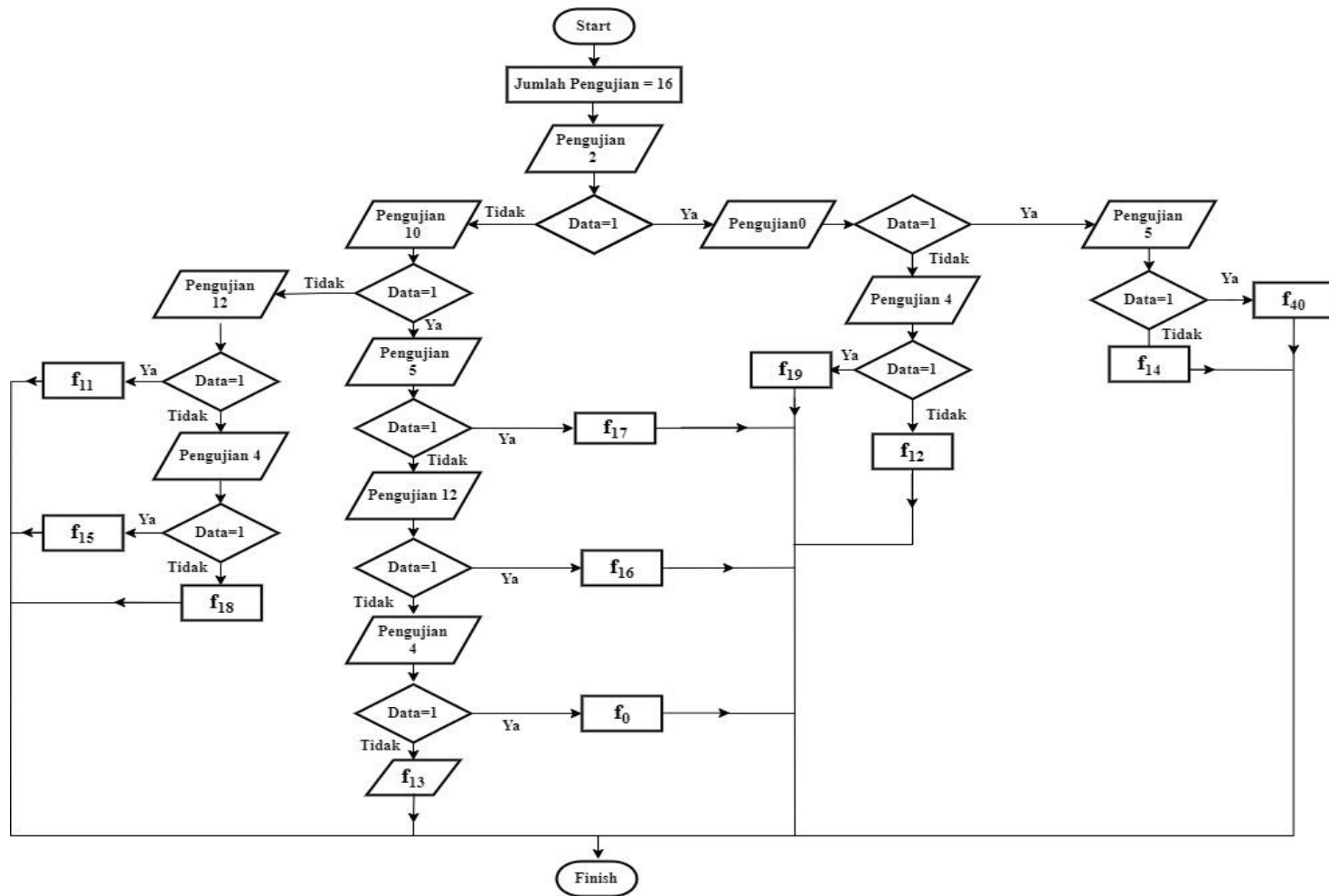
data = digitalRead(sinyalOutput);
nomorPercobaan += 'A'+String(percobaan);
output += 'B'+String(data);
outcomes = nomorPercobaan+'&'+output;
Serial.println(outcomes);
void loop() {
  if(checking){
    pulsa(percobaan[2],2);
    if(data==1){
      pulsa(percobaan[0],0);
      if(data==1){
        pulsa(percobaan[5],5);
        if(data==1){
          fault = "F30";      }
        }
      }
    }
  }
}

```



Gambar 3.29 Hasil Serial Monitor Arduino IDE Potongan 3 Rangkaian IC 74157

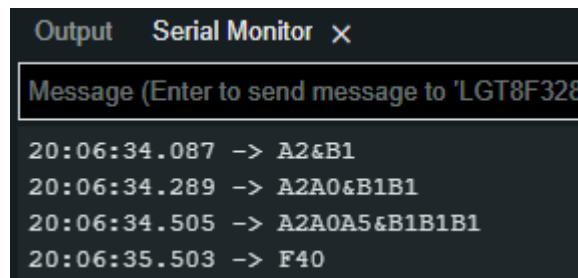
Generator set uji pada potongan 3 rangkaian IC 74157 berhasil mendeteksi kesalahan dan lokasi kesalahan sesuai dengan pohon diagnosa pada Gambar 3.19. Secara keseluruhan alur pemrograman generator set uji ke-3 bisa dilihat pada Gambar 3.30.



Gambar 3.30 Diagram Alir Coding Arduino Nano Sebagai Generator Set Uji Potongan 3 Rangkaian IC 74157

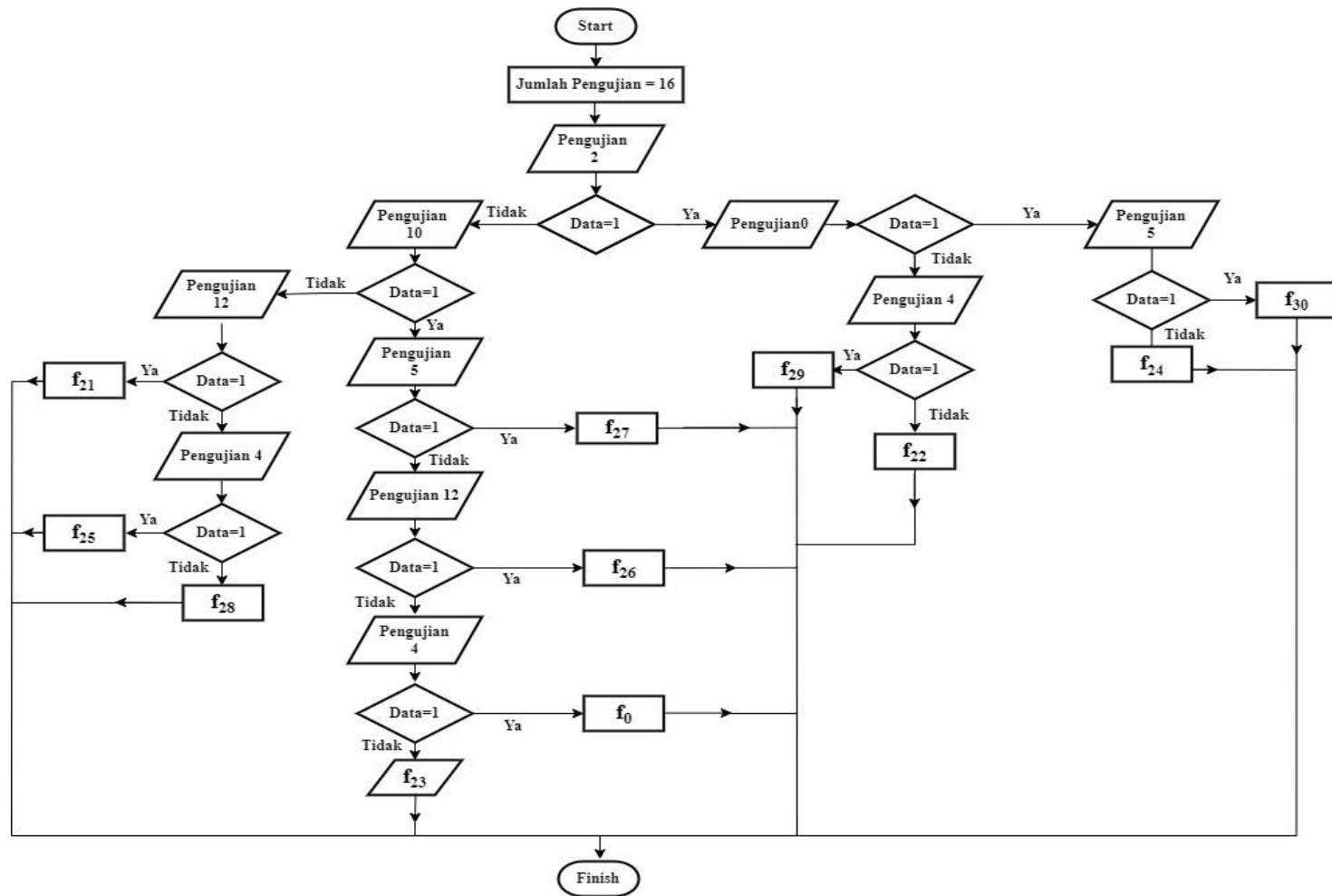
Setelah generator set uji pertama, ke-2, dan ke-3 telah berhasil mendiagnosa gangguan. Generator set uji pada potongan 4 rangkaian uji akan melakukan proses diagnosa yang dibangun menggunakan pemrograman yang sama seperti generator sebelumnya. Misalnya untuk diagnosa gangguan F₄₀ dibuat menggunakan program sebagai berikut:

```
data = digitalRead(sinyalOutput);
nomorPercobaan += 'A'+String(percobaan);
output += 'B'+String(data);
outcomes = nomorPercobaan+'&'+output;
Serial.println(outcomes);
void loop() {
  if(checking){
    pulsa(percobaan[2],2);
    if(data==1){
      pulsa(percobaan[0],0);
      if(data==1){
        pulsa(percobaan[5],5);
        if(data==1){
          fault = "F40";      }
        }
      }
    }
  }
}
```



Gambar 3.31 Hasil Serial Monitor Arduino IDE Potongan 4 Rangkaian IC 74157

Generator set uji pada potongan 4 rangkaian IC 74157 berhasil mendiagnosa gangguan sesuai dengan pohon diagnosa pada Gambar 3.20. Secara keseluruhan alur pemrograman generator set uji ke-3 bisa dilihat pada diagram alir Gambar 3.32.

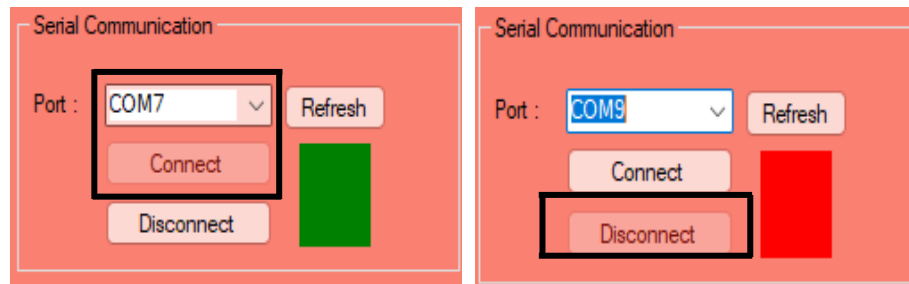


Gambar 3.32 Diagram Alir Coding Arduino Nano Sebagai Generator Set Uji Potongan 4 Rangkaian IC 74157

Pada penelitian ini hasil dari pengolahan perangkat lunak Arduino IDE diimplementasikan menjadi sebuah GUI (*graphical user interface*) yang dihasilkan menggunakan perangkat lunak Visio Studio 2022. Proses diawali dengan membuat *project form1* yang berisikan program tampilan GUI. Selanjutnya, membuat *design* tampilan yang terdiri dari *group box* 1 dan *group box* 2. Group box 1 (serial communication) tersusun dari tombol connect, tombol disconnect, label, combo box, serial port, dan panel indikator.

Tombol *connect* pada *design form1* berfungsi sebagai penghubung GUI ke port generator set uji. *Combo box* merupakan tempat pemilihan port usb generator set uji sekaligus berfungsi untuk mengaktifkan sistem IO generator. Jika generator telah terhubung dengan GUI maka panel indikator akan berwarna hijau. GUI yang telah terhubung dengan Generator bisa di lihat pada Gambar 3.33 (a). Program yang digunakan untuk memfungsikan tombol *connect* dan *combo box* adalah sebagai berikut:

```
private void btn_connect_Click(object sender, EventArgs e)    {
    if(comboBox1.Text == "COM1") {
        comboBox1.Items.Clear();
        comboBox1.Items.AddRange(SerialPort.GetPortNames());
        MessageBox.Show("Choose another port!", "Message",
        MessageBoxButtons.OK, MessageBoxIcon.Warning); }
    Else {
        timer1.Enabled = true;
        try                {
            serialPort1.PortName = comboBox1.Text;
            serialPort1.Open();
            button1.Enabled = true;        }
    }
```



a. GUI Terhubung

b. GUI *Disconnect*Gambar 3.33 *Serial Communication*

Tombol *disconnect* berfungsi untuk memutuskan sistem GUI ke generator set uji. Apabila tombol tersebut di aktifkan maka Panel indikator akan berwarna merah kembali bisa dilihat pada Gambar 3.33 (b). Panel indikator yang terhubung dengan Tombol *disconnect* dibuat dengan program sebagai berikut:

```
private void timer1_Tick(object sender, EventArgs e){
    if (serialPort1.IsOpen)
    {
        panel_indicator.BackColor = Color.Green;
        btn_connect.Enabled = false;
        btn_disconnect.Enabled = true;
    }
    else
    {
        panel_indicator.BackColor = Color.Red;
        btn_connect.Enabled = true;
    }
}
private void btn_disconnect_Click(object sender,
EventArgs e)
{
    btn_connect.Enabled = true;
    btn_disconnect.Enabled = false;
    serialPort1.Close();
    button1.Enabled = false;
}
```

Setelah sistem generator *disconnect*, langkah selanjutnya adalah mengaktifkan tombol *refresh* pada *serial communication*. Tombol tersebut berfungsi untuk memulai kembali sistem operasi generator set uji yang dibuat menggunakan program sebagai berikut:

```
private void btn_refresh_Click(object sender, EventArgs e) {
    comboBox1.Items.Clear();
    comboBox1.Items.AddRange(SerialPort.GetPortNames());
    comboBox1.SelectedIndex = 0;
}
```

Proses selanjutnya adalah membuat *group box 2* yang tersusun dari tombol *start checking*, tombol *stop checkhing*, *data grid view*, *text box*, dan tombol *next*. Ketika GUI sudah terhubung ke generator, langkah selanjutnya adalah memulai proses pengecekan kesalahan pada generator set uji dengan mengaktifkan tombol *start checkhing*. Program yang digunakan untuk memfungsikan tombol *start checkhing* adalah sebagai berikut:

```
private void button1_Click(object sender, EventArgs e) {
    timer2.Enabled = true;
    button1.Enabled = false;
    stop_checking.Enabled = true; }
}
```

Ketika program telah selesai melakukan proses pengecekan, *tool box data grid view* akan mengkonversikan nilai yang dihasilkan generator set uji menjadi sebuah tabel. Tabel data hasil pada *grid view* terdiri dari urutan pengujian, set uji, dan hasil keluaran bisa dilihat pada Gambar 3.34. Tampilan urutan pengujian dan nilai keluaran dibuat menggunakan program sebagai berikut:

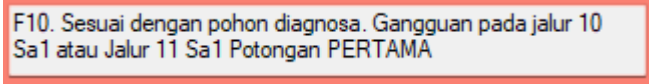
	Urutan Percobaan	Set Uji	Hasil
▶	1	2	1
	2	0	1
	3	5	1
*			

Gambar 3.34 Tampilan *Grid View*

```
string[] splitData = rawData.Split('&');
string[] percobaan = splitData[0].Split('A');
string[] output = splitData[1].Split('B');
DataSet data = new DataSet("Thesis");
DataTable table = new DataTable("IC Error Checker");
table.Columns.Add("Urutan Pengujian", typeof(int));
table.Columns.Add("Set Uji");
table.Columns.Add("Hasil");
for (int i = 0; i < output.Length; i++) {
    if (i == 0) {
        continue; }
    table.Rows.Add(i, percobaan[i], output[i]); }
```

Proses selanjutnya adalah menampilkan hasil deteksi kesalahan dan lokasi terjadinya kesalahan pada IC 74157 melalui *text box description*. Proses ini menampilkan data hasil sesuai dengan pohon diagnosa bisa dilihat pada Gambar 3.35 yang dibuat menggunakan fungsi sebagai berikut:

```
string rawData = dataSerial;
bool errorFound = rawData.Contains('F');
if (errorFound) {
    var notes = new Dictionary<string, string> {
        string error = rawData.Substring(rawData.IndexOf('F'));
        error = error.Substring(0, error.Length - 2);
        tb_description.Text = $"{error}. Sesuai dengan pohon diagnosa.
        {notes[error]}";
    };
    MessageBox.Show("Detected!", "Message", MessageBoxButtons.OK,
    MessageBoxIcon.Information);
}
```



F10. Sesuai dengan pohon diagnosa. Gangguan pada jalur 10 Sa1 atau Jalur 11 Sa1 Potongan PERTAMA

Gambar 3.35 Tampilan *Text Box Description*

Form 1 merupakan GUI potongan rangkaian IC 74157 pertama, *form 2* adalah GUI potongan rangkaian IC 74157 ke-2, *form 3* adalah GUI potongan ke-3 rangkaian IC 7415, dan *Form 4* adalah GUI potongan rangkaian IC 74157 ke-4.

Proses selanjutnya yaitu menuju ke *form 2*, *form 3*, dan *form 4* dengan cara menekan tombol *next*. Fungsi dari tombol *next* diaktifkan menggunakan program sebagai berikut:

```
private void button2_Click(object sender, EventArgs e) {
    Form1 f2,f3,f4 = new Form2,Form3,Form4();
    F2.Show();
    F3.Show();
    F4.Show();
    Visible= false;
}
```

Project form 2, *form 3*, dan *form 4* memiliki susunan komponen yang sama dengan *form 1*, hanya saja pada *form 2*, *form 3*, dan *form 4* terdapat

tombol *back*. Tombol *back* terletak di sisi kiri bagian bawah yang berfungsi untuk memindahkan GUI ke *form* sebelumnya. Fungsi pada tombol *back* tersebut dibuat menggunakan program sebagai berikut:

```
private void button3_Click(object sender, EventArgs e) {  
    Form1 f1,f2,f3 = new Form1,Form2,Form3();  
    F1.Show();  
        F2.Show();  
            F3.Show();  
Visible= false;    }
```

V. SIMPULAN DAN SARAN

5.1 Simpulan

Berdasarkan pengujian dan pembahasan dari penelitian ini, maka diperoleh kesimpulan sebagai berikut:

1. Telah berhasil dibangun generator set uji untuk mendeteksi kesalahan *stuck* pada rangkaian ic 75157 berdasarkan pohon diagnosa yang diperoleh melalui metode tabel kesalahan.
2. Efisiensi waktu pengujian generator set uji potongan rangkaian *multiplexer quad 2 line to 1 line IC 74157* adalah 68,75% sampai 81,25% dengan kedalaman pengujian tanpa terjadinya kesalahan adalah 5 level. Efisiensi pengujian pada rangkaian lengkap *multiplexer quad 2 line to 1 line IC 74157* menghasilkan efisiensi waktu pengujian yaitu 98,05% sampai 98,83% dengan pengujian terpanjang adalah 20 level.

5.2 Saran

Berdasarkan penelitian yang telah dilaksanakan, terdapat saran untuk penelitian selanjutnya, yaitu:

Pada penelitian selanjutnya, sebaiknya proses deteksi kesalahan dan lokasi terjadinya kesalahan antar generator set uji dilakukan secara otomatis dengan menggunakan GUI (*graphical user interface*) yang dapat menyatukan empat generator set uji. Sehingga hasil diagnosa kesalahan dapat langsung diketahui pada saat pengamatan, Hal ini dilakukan untuk mempercepat proses diagnosa kesalahan.

DAFTAR PUSTAKA

1. Sugiartowo. & S.T. Ambo. (2018). Simulasi Rangkaian Kombinasional Sebagai Media Pembelajaran Sistem Digital pada Fakultas Teknik Universitas Muhammadiyah Jakarta. *Seminar Nasional Sains dan Teknologi*, hal. 1-11.
2. A.H. El-Maleh., & A.K. Khaled. 2015. Simulation-Based Method for Synthesizing Soft Faults Tolerant Combinational Circuit. *IEEE Transaction on Reliability*, 1-14.
3. C. Xuebing., L. Xiao. J. Li., R. Zhang., S. Liu., J. Wang. 2018. A Layout-Based Soft Faults Vulnerability Estimation Approach for Combinational Circuits Considering Single Event Multiple Transients (SEMTs), *IEEE Transactions on Computer-Aided Design of Integrated Circuits and System*, DOI 10.1109/TCAD.2018.2834425, 1-14.
4. A.S. Repelianto. 1996. Penggunaan Metoda Tabel Kesalahan untuk Diagnosis Kesalahan Rangkaian Digital Kombinasional. Fakultas Teknik Universitas Sriwijaya, Palembang.
5. S. Roy., S.P. Milican., V.D. Agrawal. 2021. Training Neural Network for Machine Intelligence in Automatic Test Pattern Generator. *20th International Conference on Embedded Systems (VLSID) IEEE*. DOI: 10.1109/VLSID51830.2021.00059, 316-321.
6. H.F. Siregar., & M.D. Irawan. 2019. Model dan Simulasi Prototype Rangkaian Digital Konversi Gerbang AND, OR, NOT Menjadi Gerbang NAND dan NOR. *Jurnal Nasional Informatika dan Teknologi Jaringan*, 4(1), 161-166.
7. M. Raji., M.A. Sabet., B. Ghavami. 2019. Soft Faults Reliability Improvement of Digital Circuits by Exploiting a Fast Gate Sizing Scheme. *IEEE Digital Object Identifier*, 7, 66485-66495.

8. A. Baid., & A.K. Srivastava. 2013. Generating Test Pattern for Fault Detection in Combinational Corcuits Using Genetic Algorithm. *Department of Electronics and Communication Engineering, Jaypee Institute of Information Technology*, 1-4.
9. Y. Zhang., B. Zhang., V.D. Agrawal. 2014. Diagnostic Test Generation for Transition Delay Faults Using Stuck-At Fault Detection Tools. *J Electron Test*, 30, 763-780.
10. A. Kumar., & A.P Singh. 2015. Transistor Level Fault Diagnosis In Digital Circuits Using Artificial Neural Network. *Elsover Science Direct Measurement*, 82, 384-390.
11. I. Gupta. 2022. Stuck Fault Testing in Combinational Circuits Using FPGA. *Proceedings of Emerging Trends and Technologies on Intelligent Systems*, 1371, 275-284.
12. Y. Zhang., & V.D Agrawal. 2010. A Diagnostic Test Generation System. *International Test Conference*, 12(3), 1-9.
13. L. Yingchun., & Y. Liyuan. 2011. Faults Test Generation Algorithm Based on Treshold for Digital Circuit. *International Conference on Mechatronic Science, Electric Engineering and Computer*, 974-977.
14. Hoiriyah, 2020. Simulasi Gerbang Dasar Logika dalam Aplikasi. *Jurnal Teknik Informatika dan Elektro*, 2(2), 1-8.
15. Hendrata, Edward. 2018. Perhitungan Dot Vektor Dimensi Dua Input 4bit dengan Rangkaian Digital Berdasarkan Aljabar Boolean. Sekolah Teknik Elektro dan Informatika. Intitut teknologi Bandung, Bandung.
16. I. Parinduri., & S.N. Hutagalung, 2019. Perangkaian Gerbang Logika dengan Menggunakan Matlab (Simulink). *Jurnal Teknologi dan Sistem Informasi*, 5(1), 63-70.
17. C.H. Wu., K.J. Lee., S.M. Reddy. 2017. Test Generation for Open and Delay Faults in CMOS Circuits. *International Test Conference in ASIA*, 21-26.